(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 274 426 A2

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 88300074.7

(22) Date of filing: 06.01.88

(51) Int. Cl.4: G 06 F 15/40

(30) Priority: 07.01.87 US 1233

(43) Date of publication of application:
13.07.88 Bulletin 88/28

(84) Designated Contracting States: DE FR GB IT

(71) Applicant: Nikken Foods Honsha Co., Ltd.
Nikken Bldg. 305-8 1-Ban, 5-Chome, Nishinakajima
Yodogawa-Ku
Osaka (JP)

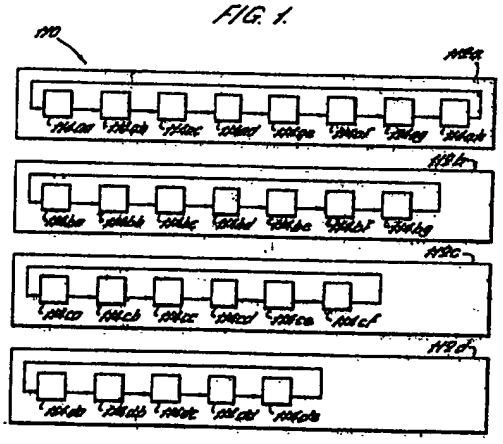
(72) Inventor: Strehler, Bernard
2235 25th Street
217 San Pedro California 90732 (US)

(74) Representative: Cross, Rupert Edward Blount et al
BOULT, WADE & TENNANT 27 Furnival Street
London EC4A 1PQ (GB)

(54) Computer memory system.

(57) A plurality of memory units (114), called "mnemonds", is each capable of storing data elements and of responding to requests to recover those data elements from storage. A plurality of mnemonds is arranged in a linear array (112), with each mnemond connected to be capable of responding to signals from its neighbor mnemonds. When a data structure is stored, each array stores a copy of that data structure, either allocating the elements of the data structure one per mnemond (with possible wrap-around to the beginning of the linear array) or allocating the data structures to one or a predetermined number of mnemonds. When a data structure is to be located in memory through cueing precise correspondences between data stored and cueing data entered identify the unique location or locations of said matching data; when a data structure is to be recovered from memory identification of its location permits recovery of data in that location or in that location and succeeding locations. One or more arrays, of differing lengths so that the allocation of data elements is also differing, is collected to form a memory bank. One or more memory banks operates under the control of a central control unit.

FIG. 1.



EP 0 274 426 A2

Description

COMPUTER MEMORY SYSTEM

The present invention relates to computer memory systems, and in particular parallel associative contents-addressable electronic memory systems.

Traditional computers and computing systems comprise at least four major subsystems: input, output, processing, and memory. The input and output subsystems are used to connect a computer to the outside world, the processing subsystem is used to perform calculations and to manipulate information, and the memory subsystem is used to record information for later use. A memory in the memory subsystem typically comprises a plurality of memory cells, each of which is used for storing one datum of the information. A processor in the processor subsystem may store ("write") data into memory by presenting that data and indicating the location of one or more cells where the data is to be stored, and may retrieve ("read") data from the memory by indicating one or more cells where the data had been stored, and accepting the data which the memory presents in response.

Sizable and complex computing systems generally work with large amounts of information, and therefore require a sizable memory with a large number of memory cells. Locating information in a sizable memory may require careful organization of data structures to be stored therein, assiduous searching of these data structures by the processing subsystem, and expenditure of much time and processing power. Accordingly, there is a need for memory subsystems in which specific stored information can be located with significantly reduced time and effort.

Related to the problem with presently available memory systems of locating information in a sizable memory is the problem of processing large amounts of data. Presently available memory systems generally allow a processor to manipulate only one data element (or only a small number of data elements) at once. When a processor works with large amounts of information, this requirement of manipulating only a small number of data elements restrains it from operating at high speeds. Paradoxically, adding more memory data elements only slows the processing operation, as the presence of more memory data elements results in the requirement of individual access and processing for each of them. There is a need for memory subsystems in which information can be manipulated in large bundles simultaneously.

As associative memory is constructed which can store data elements in a plurality of memory units ("mnemonds"). Each mnemond may be associated with neighbor mnemonds, and the mnemonds are connected by these neighbor relations to form one or more mnemond structures ("arrays"). The arrays are made to differ slightly from one another, so that data structures which are redundantly stored in multiple arrays may be overlaid with other data structures and later recovered by examination of slight differences between arrays.

The mnemonds in each may be generally arranged to form a ring-like structure, with a differing number of mnemonds in each array. A serial data structure (e.g. representing a spoken or written sentence) may be stored sequentially, one data element per mnemond, with possible wrap-around due to the ring-like structure of the arrays. A non-serial data structure (e.g. representing a picture image) may be entirely stored in a single specific mnemond of each array, or may be broken into subparts which are stored in specific sequential mnemonds. Each mnemond may be made capable of storing one or more data elements by providing the mnemond with storage indicators for each possible stored data value.

Data structures may be referred to by address values which themselves may be stored in the mnemonds which store data structures, and may themselves form an association network for allowing associative relation of data structures. Such representations of address values are redundantly stored in multiple arrays, and these stored representations of address values are made to differ slightly from one another so that they may be later recovered by examination of the slight differences between representations.

The representations of address values may generally comprise multiple indicator bits in a single register, with the numeric differences between indicator bit positions combined to form absolute address values. This representation structure allows informational tags to be attached to address values and used by a control system in conjunction with those address values. Absolute address values may be manipulated to trace and recover the associative relations of data structures.

A programmed shift register and a cycle counter registe are shown which allow a plurality of program instructions to be executed in parallel and without complex program control. Simple means are shown for sequencing and conditional branching of program steps, for "fork" and "join" process control operations, and for counting loop operations, which require only minimal hardware and software control.

Program shift registers and cycle counter registers may be generally employed to implement a parallel associative contents-addressable electronic memory system.

This invention will now be described by way of example with reference to the drawings, in which:-

Figure 1 is a block diagram of an example memory bank;

Figure 2 shows a flowchart of a method of storing data, and registers associated with that method;

Figure 3 is a block diagram of a mnemond;

Figure 3A is a block diagram of a Z-data mux;

Figure 4 shows a flowchart of a method of locating data in memory, and registers associated with that method;

Figure 5 shows a flowchart of a method of recovering data located in memory, and associated registers;

Figure 6 shows a block diagram of an example where actual retrieval of associated data is not required;

Figure 7 shows a method of associating new data with an associative key;

Figure 8 shows the structure of a memory address pointer;

Figure 9A shows the structure of a fault-tolerant variant of the AXN-register;

Figure 9B shows the structure of a fault-tolerant variant of array match circuitry;

Figure 10 is a block diagram of a control system for use with a system of banks;

Figure 11 is a block diagram of address translation hardware for a control system;

Figure 12A is a block diagram of an address extractor and demultiplexer;

Figure 12B is a block diagram of an AXOND gate;

Figure 13A is a block diagram of an X/Y-to-Z address or data translator;

Figure 13B is a block diagram of a Z-to-X/Y address or data translator;

Figure 14 is a block diagram of an address list associator;

Figure 15 is a circuit diagram of a program shift register; and

Figure 16 is a circuit diagram of a cycle counter register.

Figure 1 is a block diagram of a memory bank 110. The memory bank 110 comprises one or more arrays 112 (e.g. 112a-d), each of which comprises a plurality of mnemonds 114 (e.g. 114aa-ah 114ba-bg). Within each array, mnemonds are arranged as a linear array and connected in a ring structure, so that each mnemond 114 (e.g. 114ab) has a left-hand neighbor (e.g. 114aa), and a right-hand neighbor (e.g. 114ac). Each mnemond 114 is capable of storing data elements and of responding to requests to recover those data elements from storage.

Within each bank 110, the first array 112a comprises a number of mnemonds 114aa-ah. In this example there are eight mnemonds 114aa-ah in the first array 112a, but it will be clear to one of ordinary skill in the art that most any number of mnemonds 114 in the first array 112a will do. When multiple arrays are used, so as to permit unambiguous recovery of data stored redundantly within a bank, each succeeding array 112b-d comprises one fewer mnemonds than its preceding array--thus in this example there are seven mnemonds 114ba-bg in the second array 112b, six mnemonds 114ca-cf in the third array 112c, and five mnemonds 114da-de in the fourth array 112d.

Figure 3 is a block diagram of a mnemond 114 and its interface to bank 110 control structures. Each mnemond 114 comprises a Z-mem 310 for storing data elements. The Z-mem 310 comprises a register with a plurality of Z-bits 312. In a preferred embodiment, 96 Z-bits 312 are used, one for each printing character in the ASCII character set. The ASCII character set is a standard set of character codes which is well-known in the art. A character value is stored in the Z-mem 310 by setting the Z-bit 312 which corresponds to that character value to a logical "1". Because it comprises a plurality of Z-bits 312, a single Z-mem 310 is capable of storing representations of a plurality of character values simultaneously. However, the number of character values stored may be less than the number of character values entered, because when a given character value is stored twice or more (redundantly) in a single mnemond 114, it is still only recorded once.

Figure 2 shows a flowchart of the method of storing sequential data, and related registers. A bank 110 stores a data structure 210, comprising a set of data elements 212 (e.g 212a-z), by causing each array 112 to store a copy of that data structure 210. An array 112 (e.g 112a) stored a copy of a data structure 210 by allocating the data elements 212a-z one per mnemond 114 (e.g 114aa-ah), with the first data element 212a allocated to a selected mnemond (e.g. 114ad) and each succeeding data element (e.g. 212b) allocated to the right-hand neighbor (e.g. 114ae) of the preceding mnemond 114. Because the mnemonds 114 in each array 112 are connected in a ring structure (in a preferred embodiment of the invention), it may occur that one or more mnemonds 114 store more than one data element 212 when storage of the data structure 210 has "wrapped around" the ring structure. Because each array 112 records a copy of the data structure 210, and because each array 112 comprises a differing number of mnemonds 114, particular sets of data elements 212a-z stored by each mnemond 114 will differ in different ways 112, even though identical data structures 210 are stored therein.

Therefore, at step 220, a pseudorandom number generator ("PRNG") is used to select an initial mnemond 114 (e.g. 114ad) in each array 112. PRNGs are well-known in the art, and most any PRNG will do. In a preferred embodiment, each array is assigned a prime number, and the initial mnemond 114 in each array which is chosen for each successive sequence is simply the successive integer multiple of the prime number assigned to that array, with wrap-around when the length of that array is exceeded. The array is assigned a prime number which does not evenly divide its length.

The seed value for the PRNG is called an "address symbol". By use of a PRNG as disclosed herein, address symbols may be freely converted to sequences of individual selected mnemonds 114 in each array 112, and back. Address symbols may also be ordered such that a "next" address symbol is always calculable from a given address symbol.

At step 222, a data element 212 is stored in the selected mnemond 114 for each array 112. At step 224, the right hand neighbor of the selected mnemond 114 for each array 112 is made the new selected mnemond 114. At step 226, the data structure 210 is tested for the presence of further data elements 212. If further data elements 212 exist, control is transferred to step 222 to form a loop. Else, the method of storing data is complete, and no further steps are taken.

In a preferred embodiment of the invention, several steps are taken to optimize this storage mechanism for text character strings. Text character strings usually concentrate most of their character values in a few common characters, exacerbating the redundancy noted above by causing unequal use of character storage locations. To compensate for this, and thereby cause approximately equal use of all character storage locations within a single mnemond, when a character string is stored in an array 112, each succeeding character value may be offset one more than the preceding character value, so that the redundancy noted above has reduced effect. One particular common character is the space, which is treated specially--it is simply not stored at all.

For example, to store the character string "TO BE OR NOT TO BE", the character value for "T" would be stored in the first mnemond 114aa used used in each array, the character value for "P" ("O" offset by one) would be stored in the second mnemond 114ab used, the character value for "E" ("B" offset by three) would be stored in the fourth mnemond 114ad used, the character value for "I" ("E" offset by four) would be stored in the fifth mnemond 114ae used, and so on until the character value for "W" ("E" offset by 18) would be stored in the last mnemond 114ac used, after a double wrap-around.

For certain data, it may be desirable to store, rather than a single copy of the data, a pattern of copies of the data which differ but which are closely related. An exemple of such a pattern may be found in storage of visual and voice data for later retrieval, as visual and voice data may be subject to significant variation in replication by a source, as well as significant error in sensing by a sensor device. Other and further information about sensing and replication of visual and voice data may be found in European Patent Application No. (Attorney reference 31598/000).

To effect retrieval of stored data structures 210 which are close but not exact matches of cue structures 410, two clear alternatives are presented. A first alternative is to allow a match to be recognized where the cue structure 410 is closely related to a stored data structure 210, but is not necessarily an exact match. For example, a mnemond 114 may declare a mnemond match when a cue element 412 matches a data element 212 stored in that mnemond 114 or matches a data element 212 stored in a neighbor mnemond 114. A second alternative is to store a pattern of data structures 210 such that a closely related cue structure 410 will be exactly replicated in the pattern and will exactly match one of the stored data structures 210.

A variation of the first alternative is applied when matching cue structures 412 or data structures 212 which include space characters. As noted above, in a preferred embodiment, the space character is not stored is any mnemond 114 and is never a data element 212. Accordingly, when a cue structure 410 is presented which would ordinarily include a space character, the space character is replaced with a "wild card" character which is matched by all mnemonds 114. Thus a cue structure 410 including one or more space characters will not fail to match a stored data structure 210 due to the lack of space characters stored as data elements 212.

While the structure of a mnemond 114 could easily be altered to accept matches which are not quite exact, in an embodiment where matches with closely related cues are desired the second alternative above is chosen. It is preferred to store each such closely related pattern as a stored data structure 210 and to require that each mnemond 114 make an exact match. This simplifies the hardware components of each mnemond 114 and allows the exactness of the match to be easily and flexibly altered by suitable tailoring of parameters for control software which may decide what data structures 210 to actually store in each bank 110. Each stored data structure 210 may also have a "preference" weighting assigned to it by inclusion of a "preference" value in a stored memory address pointer 216, as further disclosed herein with reference to Figure 8.

Figure 4 shows a flowchart of the method of locating data in memory, and related registers. A bank 110 locates a data structure 210, previously stored in memory, by presenting a cue structure 410, comprising a set of cue elements 412 (e.g. 412a-z), simultaneously to each array 112 for matching. The cue structure 410 is successfully detected and its location identified if each array 112 declares an array match (i.e. a match between the cue structure 410 and data stored in that array 112).

An array 112 (e.g. 112a) locates a data structure 210, previously stored in memory, by presenting each succeeding cue element 412 to all its mnemonds 114 at once, allowing each mnemond 114 to compare the cue element 412 with its Z-mem 310. Each mnemond 114 (e.g. 114ad) declares a mnemond match (i.e., a match between a data element 212 and data stored in that mnemond 114) if and only if the cue element 412 (e.g. 412b) is present in its Z-mem 310 and either (1) this is the first cue element 412a, or (2) its left-hand neighbor (e.g. 114ac) declared a mnemond match for the preceding cue element 412 (e.g. 412a). An array 112 declares an array match if and only if a single mnemond 114 declares a mnemond match for the last cue element 412 of the cue structure 410. Then, if all arrays 112 in a bank 110 declare an array match, the cue structure 410 is successfully detected as being present in memory and its location is simultaneously identified.

Note that a data element 212 may comprise one or more data points 220, i.e. Z-bits 312 which are set to a logic "1" in a Z-mem 310. More than one data point 220 per data element 212 is generally envisioned for picture data rather than ASCII text. Similarly, a cue element 412 may comprise one or more cue points 418. Each cue point 418 must match a data point 220 for a mnemond match.

Therefore, at step 420, each array 112 has an AXN-register 414 (shown in Figure 3), comprising a plurality of AXN-bits 416 (shown in Figure 3), one per mnemond 114, and clears all the AXN-bits 416 to a logical "0". At step 422, the first cue element 412 (e.g. 412a) is presented to each mnemond 114 in the array 112 simultaneously, and each mnemond 114 compares the cue element 412 with its Z-mem 310, declaring a mnemond match if the cue element 412 is present in its Z-mem 310. Each mnemond 114 which declares a mnemond match sets its AXN-bit 416 to a logical "1".

At step 424, the cue structure is tested for the presence of further cue elements 412. If further cue elements 412 exist, control is transferred to step 426 to start a loop. Else control is transferred to step 430. At step 426, each array 112 shifts the AXN-register 414 rightward one bit, thus shifting the AXN-bit 416 associated with each mnemond to the AXN-bit 416 immediately to its right in the AXN-register 414. At step 428 the next cue element 412 is presented to all mnemonds 114 in the array 112 whose corresponding AXN-bit is in the "1" state 416 and each such mnemond 114 compares the cue element 412 with its Z-mem 310, declaring a mnemond match if the cue element is present in its Z-mem 310. Each such mnemond 114 which fails to declare a mnemond match clears its AXN-bit 416 to a logical "0"; else the AXN-bit 416 for that mnemond 114 remains set to a logical "1". Control is then transferred to step 424 to complete the loop.

At step 430, each array 112 declares an array match if and only if a single AXN-bit 416 in its AXN-register 414 remains set to a logical "1"; this indicates that the complete cue structure 410 is present in the mnemonds which comprise that array 112. At step 432, the bank 110 declares a bank match if and only if all its arrays 112 declare an array match. The method of locating data in memory is then complete, and no further steps are taken.

Once the cueing structure 410 is matched by each array 112 in a bank 110 (causing the bank 110 to declare a bank match), a stored data structure 210 which contains that cueing structure 410 as a substring has been detected. In a preferred embodiment, the stored data structure 210 may include, at its end, an end of structure mark 214 and a memory address pointer 216 to its starting location, and at its start, a start of structure mark 218. The memory address pointer 216 points to the beginning of the stored data structure 210 and thus allows the entire stored data structure 210 to be retrieved. In an alternate embodiment, the memory address pointer 216 may point to an indirect table, to an address in conventional memory, or to some other location to facilitate reference to the entire stored data structure 210.

Thus, once a terminal substring of the stored data structure 210 has been located, the end of structure mark 214 and the memory address pointer 216 can be retrieved in addition to the terminal data elements 412 of the data structure 210. At the completion of the method of locating data in memory (disclosed with reference to Figure 4), the location of AXN-bits 416 set to a logical "1" corresponds, in each array, to the location of the next (uncued) data element 412 stored in the bank 110 as part of a single data structure 410. The terminal data elements are associated with the cue structure 414 previously presented, just as data is associated with an associative key in a contents-addressable memory. These terminal data elements 412 may then be retrieved from memory, as described with reference to Figure 5.

Figure 5 shows a flowchart of the method for retrieval of the terminal part of a stored data structure 210. Because each array 112 may store more than one stored data structure 210, mnemonds 114 which follow the cueing structure 410 in each array 112 may contain values which do not unambiguously indicate their data elements 212, the end of structure mark 214, or the memory address pointer 216. In general, mnemonds 114 in multiple arrays 112 must be examined (e.g. by bitwise ANDing their contents) to disambiguate these values and thus detect the presence of common stored data elements 212, the end of structure mark 214, the memory address pointer 216, and the start of structure mark 218. However, due to impossible spurious combinations of data, values may still lead to detection of ambiguous stored data elements 212; these ambiguous stored data elements 212 may be disambiguated through their context.

Therefore, at step 520, the AXN-register 414 is shifted one position to the right in each of the arrays 112 in that bank 110. At step 522, the contents of the Z-mem 210 of all mnemonds 114 whose AXN-bit 416 is in a logical "1" are retrieved; these contents are bitwise ANDed with each other and the result stored in an output Q-buf 344 (shown in Figure 3), whose structure is that of a Z-mem 310. At step 524 the contents of the output Q-buf 344 is examined for the presence of an end of structure mark 214. If an end of structure mark 214 is detected, control is passed to step 526, else the recovered data element 212 is output for processing, after which control is returned to step 520. At step 526, the content of the A-mem 220 for that mnemond 114 is recovered, and output for processing. The method of retrieving data from memory is then complete, and no further steps are taken.

Because stored data is located in memory and retrieved from memory based on its contents (i.e. the memory system is a contents-addressable and "associative" memory), it is desirable to prevent storage of an associative key (i.e. a data structure 210) in two different sets of mnemonds 114 in memory. Therefore, before actual storage of any data in memory takes place, a cueing search is performed to determine that the same data is not already present in memory. The cueing search is further disclosed with reference to Figure 4. If the associative key is already present in memory, storage does not proceed normally; instead the new data is associated with the previously-stored associative key. Associating new data with an associative key is further disclosed with reference to Figure 7.

It will be clear to one of ordinary skill in the art that there are two distinct operations which are disclosed above. First, an associative key can be located in memory, i.e. its presence can be detected and its location identified: once the cueing structure 410 is matched by each array in a bank, its presence has been detected and its location has been identified. Second, data which is associated with that key can be retrieved from memory: once the location of the cueing structure 410 has been identified, the data which follows that cueing structure 410 in a stored data structure 210 is associated with the cueing structure 410 and can be retrieved. Recovery of the memory address pointer 216, and thus the entire stored data structure 210, generally depends upon location and retrieval of the terminal portion of the stored data structure 210. It will also be clear to one of ordinary skill in the art that the redundancy of multiple arrays 112 is only necessary when the cueing structure

410 is not a terminal substring of the stored data structure 210, i.e. only for the retrieval operation, and that where retrieval of associated data stored in a contents-addressable memory is not required, the data need not be stored redundantly and multiple arrays are not needed. Figure 6 shows a block diagram of an example embodiment where actual retrieval of associated data is not required.

The system 610 may comprise an input sensor 612 which provides input data on a data bus 614, an associative memory 616 as disclosed herein, an ordinary memory 618 which records a table of locations where the input data has been stored in the memory system 616, and a processor element 620. Input data is transmitted on the data bus 614 from the input sensor 612 to the associative memory 616 for location in memory (without retrieval). The associative memory 616 responds by transmitting to the processor element 620, on the data bus 614, a value indicating a location where the input data has been found. The processor element 620, which may comprise a microprocessor of any standard make, then recovers the data from the ordinary memory 618, using the location provided by the associative memory 616 as an index into the table of locations of the ordinary memory 618. Because only one block of associated information is stored in association with each associative key, there is no need to retrieve that data element from the associative memory 616, and thus there is no need to include multiple arrays in the associative memory 616.

As disclosed with reference to Figure 2, a data structure 210 comprises one or more data elements 212, followed by an end of structure mark 214 and preceded by a start of structure mark 218. The data elements 212, an end of structure mark 214, and a start of structure mark 218 are stored in the Z-mems 310 of one or more mnemonds 114. In a preferred embodiment, a memory address pointer 216 is stored in the A-mem 314 of the mnemond 114 which stores the end of structure mark 214 (or one or more mnemonds 114 following), and this memory address pointer 216 refers back to the beginning of the data structure 210 (i.e. a memory address pointer 216 to the start of structure mark 218 in each array 112 where this data structure 210 is stored).

As disclosed with reference to Figure 6, part of a data structure 210, previously stored in memory, can be used to reference other parts of the same data structure 210 in an associative manner. Thus for example, if multiple data structures 210 representing spoken lines from Shakespearean plays were stored in memory, one could use the first half (or any other part) of any line as a cue structure 410 and recover enough information to retrieve the entire data structure 210.

Figure 7 shows the method of associating new data with an associative key. It is often desirable to associate data structures 210 with each other in a more flexible manner, thus making an association which it is possible to add to at another time. In a preferred embodiment of the invention, one or more memory address pointers 216 will also be stored in the A-mem 314 of the mnemond 114 which stores the start of structure mark 218. These memory address pointers 216 point to other data structures 210 (e.g. 210b) which are associated with the data structure 210 (e.g 210a) they are stored as a part of. Thus for example, when recovering the memory address pointer 216 of a data structure 210 which represents a spoken line from a Shakespearean play, one might also be able to recover (if an appropriate associative memory address pointer 216 has been stored therein) a data structure 210 which represents commentary on that line or information about its meaning.

Figure 8 shows the structure of a preferred embodiment of a memory address pointer 216. Because there may be many mnemonds 114 and thus many possible addresses to distinguish, it is generally impractical to allocate one A-bit 316 of an A-mem 314 per possible address. Rather than this one bit storage code, memory address pointers 216 are stored in a three (or more) bit storage code which compacts the number of A-bits 316 required per A-mem 314 greatly, but which still allows storage of multiple memory address pointers 216 in a single A-mem 314 without losing the ability to recovery each individual memory address pointer 216 at a later time.

Each memory address pointer 216 comprises a set of three (or more) indicator bits 802 which are set to a logic "1" in a register 804 whose bits 806 are generally set to a logic "0". It will be clear to one of ordinary skill in the art that this situation could be reversed, and that a memory address pointer 216 could comprise indicator bits 802 set to a logic "0" in a register 804 whose bits 806 are generally set to a logic "1", or could comprise indicator pairs of bits or other recognizable indicator patterns of bits. The offset from each indicator bit 802 to the next indicator bit 802 determines a numeric value 808, which in a preferred embodiment is simply the difference between bit positions of the indicator bits. These numeric values 808 are combined in a predetermined manner to comprise a combined numeric address 810.

Thus, where there are exactly three indicator bits 802, there are two numeric values 808 which are determined from the indicator bits 802. In a preferred embodiment these two numeric values 808 are combined linearly, thus:

(combined numeric address 810) =
1000 * (first numeric value 808)
+ 1 * (second numeric value 808)

It will be clear to one of ordinary skill in the art that other linear combinations of the numeric values 808 could be chosen, that more than three indicator bits 802 and thus more than two numeric values 808 could be used, and that some numeric values 808 could be ignored (i.e. their contribution would be zero) when calculating the combined numeric address 810.

In a preferred embodiment, more than three indicator bits 802 may be used, but only the first three count towards determining a combined numeric address 810. This allows numeric values 808 to be combined with a memory address pointer 216 which are useful informational tags when using that memory address pointer 216 to retrieve a data structure 210 that it refers to. For example, in a preferred embodiment, one or more numeric

values 808 may be used to represent the "quality" of the associative link, and one or more numeric values 808 may be used to represent the "abstraction level" of the associative link.

It will be clear to one of ordinary skill in the art that, after storage of more than one memory address pointer 216 in a given mnemond 114, there will be some ambiguity introduced by the existence of combinations of indicator bits 802 which are not attributable to any memory address pointer 216 originally stored in that A-mem 314. So that it is possible to recover each individual memory address pointer 216 from the A-mem 314 in which it is stored at a later time, it is desirable to introduce some differences between the storage of the same memory address pointer 216 in differing arrays 112, and to introduce some differences between the storage of differing memory address pointers 126 in the same mnemond 114. These differences can then be used to recover the individual memory address pointers 216 which were stored, much as it is possible to recover the individual data elements 212 which were stored in each mnemond 114 and data structures 210 which were stored in each bank 110, due to the differences between the storage of the same data structure 210 in differing arrays 112.

The present system introduces these differences by storing multiple memory address pointers 216, with consequent multiple different indicator bits 802, at differing offsets from the beginning offset of the original memory address pointer 216 stored in that A-mem 314. An initial offset 812, the offset which the first indicator bit 802 has from the end of the register 604, is chosen which differs for representations of the same memory address pointer 216 stored in differing arrays 112, and which differs for representations of differing memory address pointers 216 stored in the same mnemond 114. In a preferred embodiment, this may be achieved by use of a formula substantially as shown:

(initial offset 812) =
k0 +
k1 * (array 112 number) +
k2 * (memory address pointer 216 number)

Thus for example, if k0 = 1, k1 = 2, and k2 = 3, the first memory address pointer 216 would be stored at an initial offset 812 of six in the first array 112, eight in the second array 112, and ten in the third array; the second memory address pointer 216 would be stored at an initial offset 812 of nine in the first array 112, eleven in the second array 112, and thirteen in the third array.

It will be clear to one of ordinary skill in the art how these differing offsets allow recovery of the originally stored memory address pointer 216. When reading a memory address pointer 216 from a set of A-mems 314, a memory system controller must offset the A-bits 316 by the opposite of the amount they were originally offset when stored in that A-mem 314.

Figure 9A shows the structure of a fault-tolerant variant embodiment of the AXN-register 414. Because it may occur that one or more mnemonds 114 may fail during operation or exhibit errors due to mistakes in manufacture, it is desirable to provide a mechanism for removing the failed mnemond 114 from the array 112 of which it is a part. Accordingly, test and disabling circuitry is provided. XOR gate 902 compares the actual output from the mnemond 114 with its expected output. The output of XOR gate 902, on line 904, is propagated through AND gate 906 when a "TEST" signal on line 908 is set to a logic "1". The output of AND gate 906, on line 910, is preserved via a latch 912 and propagated to modified circuitry for each AXN-bit 416, to disable that AXN-bit 416 if a fault is detected for its corresponding mnemond 114.

The disable signal, on line 914, is propagated to the select input 916 of a multiplexor 918, and determines which of the two input bits 920a-b are propagated by the multiplexor 918 to the next AXN-bit 416. Input to the multiplexor 918 are the AXN-bit 416 corresponding to this particular mnemond 114 and the AXN-bit 416 immediately preceding. Thus, when the disable signal is set to a logic "1", the AXN-bit 416 corresponding to the preceding mnemond 114 will be propagated to the AXN-bit 416 for the following mnemond 114, without interference by the mnemond 114 which is disabled.

Figure 9B shows the structure of a fault-tolerant variant of the array match circuitry. Because entire arrays 112 are also subject to failure, it is desirable to provide circuitry which allows a bank 110 to declare a bank match, even when one of its arrays 112 has failed. Accordingly, AND gates 952, one per array 112 in the bank 110, each receives inputs on lines 954 from each array 112, indicating for each array 112 if that array 112 has declared an array match. The output of all AND gates 952 are propagated to circuitry 958 which, as is well known in the art, will indicate a bank match signal on line 956 if and only if at least all-but-one of the arrays 112 have declared array matches.

Figure 15 is a circuit diagram of a program shift register ("PSR") 1510. This circuit is used to implement certain program functions, although it will be clear to one skilled in the art that the use of a PSR 1510 for any given function is not required, and may be replaced by a microprocessor operating under the control of suitable software, by a programmed logic array, or by other similar structures. A PSR 1510 is usually preferred because they are inexpensive to construct, particularly because many of them may operate in parallel.

A parallel/serial shift register 1512, which may comprise one or more standard 7496-type shift registers in series, is arranged with a parallel input for its first bit 1514a electrically connected to a voltage source 1516 (which indicates a logic "1") and with parallel inputs for each of its other bits 1514b-z similarly electrically connected to ground 1518 (which indicated a logic "0"). A parallel load input 1520 and a clear input 1522 for the shift register 1512 are electrically connected to an input trigger line 1524. Thus, when a logic "1" is transmitted on the input trigger 1524, the shift register 1512 is cleared and parallel loaded with a logic "1" bit in its first bit 1514a and logic "0" bits in all its other bits 1514b-z.

Each bit 1514 (e.g. 1514a) has its value transmitted on a bit output line 1526 (e.g. 1526a) directly to intermediate circuitry and [illegible]ately to the input or an OR gate 1528 (called [illegible] program advance OR" or a "PAOR"). The output of this OR gate is transmitted on line 1530 to the input of an AND gate 1532. A clock signal on line 1534 is also input to the AND gate 1532, and the output of the AND gate 1532 is transmitted to a clock input 1536 for the shift register 1512. Thus, when the logic "1" which was initially loaded into the first bit 1514a appears on a bit output line 1526 (e.g. 1526a), it causes line 1530 to be set to a logic "1" and thus causes the shift register 1512 to be enabled for shifting when the clock signal on line 1534 is triggered.

A PSR 1510 is called a "program" shift register 1510 because bits 1514 may be manipulated in several ways by intermediate circuitry, as mentioned above, before their values are transmitted to the OR gate 1528. For example, they may also have their values transmitted on a bit output line 1526 to trigger the initiation of another PSR 1510 (i.e., a "fork" process control operation, as is well-known in the art). This is indicated in the figures by naming a signal which the bit 1514 triggers, e.g. "A". In a preferred embodiment, the relevant bit output line 1526 may be electrically connected to the input trigger line 1524 of another PSR 1510.

They may also have their values manipulated by logic circuitry, so as to await one or more other signals before the PSR 1510 is advanced (i.e., a "join" process control operation, as is well-known in the art). This is indicated in the figures by a plus sign followed by the signal name which is awaited, e.g. "+B". For example, an AND gate 1538 may cause the PSR 1510 to await the "B" signal on line 1540 before it is advanced by the clock signal on line 1534. For another example, a pair of AND gates 1542 and 1544 may similarly cause the PSR 1510 to await either the "C" signal on line 1546 or the "D" signal on line 1548 before it is advanced by the clock signal on line 1534. Other similar intermediate circuitry for controlling the program advance of a PSR 1510 will be clear to those of ordinary skill in the art.

Figure 16 is a circuit diagram of a cycle counter register ("CCR") 1610. Like the PSR 1510, this circuit is also commonly used throughout a preferred embodiment of the present invention to implement certain program functions, although it will be clear to one skilled in the art that its use is not required if it is replaced by a microprocessor operating under the control of suitable software, by a programmed logic array, or by other similar structures. Like the PSR 1510, a CCR 1610 is often preferred because they are inexpensive to construct, and because many of them may operate in parallel.

A parallel/serial shift register 1612, which may comprise one or more standard 7496-type shift registers connected in series, is arranged with a parallel input for its first bit 1614a electrically connected to a voltage source 1516 (which indicates a logic "1"), and with parallel inputs for each of its other bits 1614b-z similarly electrically connected to ground 1518 (which indicates a logic "0"). A parallel load input 1620 and a clear input 1622 for the shift register 1612 are electrically connected to an input trigger line 1624. Thus, like the PSR 1510, when a logic "1" is transmitted on the input line 1624, the shift register 1612 is cleared and parallel loaded with a logic "1" bit in its first bit 1614a and logic "0" bits in all its other bits 1614b-z.

The first bit 1614a has its value transmitted on a bit output line 1626, negated by an inverter 1628, and then transmitted on a line 1630 to the input of an AND gate 1632. A clock signal on line 1634 is also input to the AND gate 1632, and the output of the AND gate 1632 is transmitted on line 1635 to an OR gate 1636. A count trigger signal on line 1638 is also input to the OR gate 1636, and the output of the OR gate 1636 is transmitted to a clock input 1640 for the shift register 1612. Thus, when the logic "1" which was initially loaded into the first bit 1614a appears on a bit output line 1626a, it causes line 1632 to be set to a logic "0" and thus causes the shift register 1612 to be disabled for shifting when the clock signal on line 1634 is triggered. But when the trigger signal on line 1638 is set to logic "1", the CCR 1610 shifts repeatedly, once per trigger of the clock signal on line 1634, until the logic "1" is returned to the first bit 1614a via bit output line 1626z.

Figure 3, a block diagram of a mnemond 114, also shows the registers which are used to implement the mnemond 114 procedures. Each mnemond 114 may also comprise a A-mem 314, which itself comprises a register with a plurality of A-bits 316. Like the Z-men 310, which is used for storing data, the A-mem 314 is used for storing address pointers to data. Each mnemond 114 records address pointers to data in its A-mem 314 to make associations between cue structures (e.g., text strings which serve as keys to locate information) and data structures. Each data structure contains a special end of structure mark 214 which indicates the end of that data structure 210. In a preferred embodiment, the mnemond 114 which records the end of structure mark 214 also records in its A-men 314 an address pointer to the beginning of the data structure 210. This allows the entire data structure 210 to be located when only a part of the data structure 210 is matched by a cue structure 410.

Each mnemond 114 may also record address pointers to data in its A-men 314 at the beginning of each data structure 210, to represent an association between that data structure 210 and other associated data. The A-men 314 contains the address pointer to the associated data, plus other property information about the associated area, such as a value to indicate the level of abstraction at which the association is made.

Each mnemond 114 also comprises a match-ctr 318, which itself comprises a register with a plurality of match-bits 320. The match-ctr 318 is used to record the number of cue "data points" for which a match has been detected. Where a serial data structure 210 (e.g. representing a spoken or written sentence) is cued, normally each mnemond 114 will record exactly one data element 212, but where a non-serial data structure 210 (e.g. representing a picture image) is cued, each mnemond 114 may record a plurality of data elements 212 at once. Each of these data elements 212, when cued for a single mnemond 114, is called a "data point".

The Z-mem 310, A-mem 314, and match-ctr 318 for each mnemond 114 are respectively associated with registers comprising the array 112 which contains that mnemond 114, and with registers comprising the bank

110 which contains that array 112. These associated registers and their connections with the registers which comprise each mnemond 114 are also shown in Figure 3.

Z-mems 310 are all associated with a single Q-buf 322 per bank 110, itself comprising a plurality of Q-bits 344 in a structure identical to a Z-mem 310, and a single Z-CCR 324 per bank 110, itself comprising a CCR 1610 equal in length to a Z-mem 310. The Q-buf 322 records one or more data elements 212 which the bank 110 has buffered for input/output operations. The Z-CCR 324 is a counting register which enables the bank 110 to direct a complete cycle of readin/readout of the Z-mem 310 and of the Q-buf 322.

In a preferred embodiment, associated with the A-mem 314 is an A-buf 326, itself comprising a structure identical to a A-mem 314, and an A-CCR 328, itself comprising a CCR 1610 equal in length to an A-mem 314. Like the Q-buf 322, the A-buf 326 records one or more address pointers which the bank 110 has buffered for storage or retrieval operations. Like the Z-CCR 324, the A-CCR 328 is a counting register which enables the bank 110 to direct a complete cycle of readin/readout of the A-mem 314 and of the A-buf 326.

Associated with the match-ctr 318 is a Q-ctr 330, itself comprising a structure identical to a match-ctr 318, and a match-CCR 322, itself comprising a CCR 1610 equal in length to a match-ctr 318. The Q-ctr 330 counts the number of data points which the bank 110 has presented to its mnemonds 114 for cueing. Like the Z-CCR 324, the match-CCR 332 is a counting register which enables the bank 110 to direct a complete cycle of readin/readout of the match-ctr 318 and of the Q-ctr 330.

Each mnemond 114 also comprises a FIRST register 334, which itself comprises a latch with a single bit value 336. The single bit value 336 of the FIRST register 334 is a flag bit indicating that the mnemond 114 is responding to the first cue element 412a of a cue structure 410. Because the flag bit of the FIRST register 334 can be set when the first cue element 412a is being presented, and it will be cleared automatically when later cue elements are presented.

Each mnemond 114 also comprises a MATCH register 338, which itself comprises a latch with a single bit value 340. The single bit value 340 of the MATCH register 338 is a flag bit indicating that the mnemond 114 has declared a "mnemond match" (i.e. a match between a data element 212 stored in the Zmem 310 for this mnemond 114 and the cue element 412 which was presented. The MATCH register 338 must be reset for each mnemond 114 for each cue element 412 which is presented to that mnemond 114.

As disclosed with reference to Figure 4, each array 112 comprises an AXN-register 414, which itself comprises a shift register with one AXN-bit 416 per mnemond 114 in that array 112. Each mnemond 114 comprises an AXN-latch 342 which records the value of the AXN-bit 416 associated with that mnemond 114 while the AXN-register 414 is shifted. As disclosed with reference to Figure 4, the AXN-register 414 is shifted right one bit after each cue element 412 is presented by the bank 110 to its mnemonds 114 (during a cueing operation), after each data element 212 is presented by the bank 110 to its mnemonds 114 (during a storing operation), and after each data element 212 is retrieved by the bank 110 from its mnemonds 114 (during a retrieval operation).

Each bank 110 also comprises a plurality of PSRs 1510 for controlling the operation of the bank 110 registers (Q-buf 322, Z-CCR 324, A-buf 326, A-CCR 328, Q-ctr 330, and match-CCR 332), array 112 registers (AXN-register 414), and mnemond 114 registers (Z-mem 310, A-mem 314, match-ctr 318, FIRST register 334, MATCH register 338, and AXN-latch 342). In Figure 3, each PSR 1510 is labelled with program codes as disclosed with reference to Figure 15, thus it should be clear to one of ordinary skill in the art how the store, locate, and retrieve operations are implemented.

Each CCR 1510 (Z-CCR 324, A-CCR 328, and match-CCR 332) has associated with it a "start" signal ("B" on line 324a, "K" on line 328a, and "A" on line 332a respectively) and an "end" signal ("X" on line 324b, "Z" on line 328b, and "W" on line 332b respectively). A single "reset" signal on line 324c suffices to reset all three CCRs 1510 simultaneously. Because the connection of clock inputs is disclosed in detail with respect to Figures 15-16, clock inputs are generally not shown in Figure 3 and may be presumed to be tied to a common system clock signal except where otherwise specified.

Associated with the Z-CCR 324 are the Z-mem 310 and Q-buf 322 registers; these latter two registers are clocked by the same signal on line 324a as the Z-CCR 324. Thus, when a PSR 1510 triggers the signal on line 324a, the Z-CCR 324 is made to count its entire cycle, while the Z-mem 310 and the Q-buf 322 count their entire cycles simultaneously. This allows individual Z-bits 312 in the Z-mem 310 and their corresponding Q-bits 344 in the Q-buf 322 to be directly compared and directly transferred between the two registers.

In like manner, associated with the A-CCR 328 are the A-mem 314 and the A-buf 326 registers, and associated with the match-CCR 332 are the match-ctr 318 and Q-ctr 330 registers. The A-CCR 328, A-mem 314, and A-buf 326 registers are all triggered by a signal on a single line 328a. The match-CCR 332 is triggered by a signal on a line 332a; this signal is propagated through OR gate 318a to trigger the match-ctr 318 or through OR gate 330a to trigger the Q-ctr 330.

OR gates 318a and 330a serve to allow other signals to also trigger the match-ctr 318 and the Q-ctr 330. The Q-ctr 330 is also triggered by Q-bits 344 which are set to a logic "1", thus each data point stored in the Q-buf 322 and presented to the mnemond 114 for cueing causes the Q-ctr 330 to advance on position (since it is a CCR 1610). The match-ctr 318, in contrast, is triggered by the output of AND gate 318b, which compares Q-bits 344 with their corresponding Z-bits 312. Thus the match-ctr 318 will advance only when the cued data point from the Q-buf 322 matches a stored data point from the Z-mem 310. When the match-ctr 318 and the Q-ctr 330 are later cycled by the signal to the match-CCR 332 on line 332a, AND gate 338a will output a logic "1" if and only if an equal number of data points were cued and were detected.

The FIRST register 334 is also triggered by the signal to Z-CCR on line 324a, thus it is initiated with a logic "1" when the first cue element [redacted] is presented and cycles to a logic "0" for all succeeding cue elements 412. The FIRST register 334 and the MATCH register 338 supply inputs to AND gate 348a, while the MATCH register 338 and the AXN-latch 342 supply inputs to AND gate 348b; AND gates 348a-b supply inputs to OR gate 348c, which supplies the parallel input to the AXN-bit 416 of the AXN-register 414 associated with that mnemond 114. The input of the AXN-latch 342 is supplied by the AXN-bit 416.

Thus, the AXN-latch 342 for each mnemond 114 stores the previous AXN-bit 416 from the AXN-register 414 when the AXN-register 414 is shifted and cleared, and supplies the next AXN-bit 416 (after propagation through gates 348a-c) when the AXN-register 414 is reloaded. The gates 348a-c provide that each AXN-bit 416 is set to a logic "1" in only two cases: (1) when the first cue element 412a has been matched by this mnemond 114, or (2) when the preceding mnemond 114 for this mnemond 114 has had its AXN-bit 416 set to a logic "1", and a cue element 412 other than the first cue element 412a has been matched by this mnemond 114.

The Z-mem has its data input port 310a supplied with data from a Z-data mux 350. The Z-data mux 350 is further disclosed with respect to Figure 3A; it is shown there to comprise five gates 350a-e responsive to five input signals on lines 352a-e. The nature and use of the five input signals will be clear to one of ordinary skill in the art from an examination of the circuit shown in Figure 3A.

A CLEAR signal on line 352a may be set by the bank 110 to clear Z-bits 312, in a particular Z-mem 310 or in all Z-mems 310, to a logic "0". If the CLEAR signal is not set, the old Z-bit 312 is input on line 352b (from the output of the Z-mem 310) and recycled back into the Z-mem 310, unless a new data element 212 is directed to be stored in the Z-mem 310. A STORE-ENABLE signal on line 352c may be set by the bank 110 to enable storing of a new Z-bit 312. If the STORE-ENABLE signal is set, a new Z-bit 312 on line 352d (from the output of the Q-buf 322) is stored in each mnemond 114 whose AXN-bit 416, input on line 352e, has been set to a logic "1".

The A-mem 314 has its data input port 314a similarly supplied with data from an A-data mux 354. The A-data mux 354 is almost identical in structure to the Z-data mux 350; differing only in that it is responsive to an old A-bit 316 rather than an old Z-bit 312 (from the output of the A-mem 314) and responsive to a new A-bit 316 rather than a new Z-bit 312 (from the output of the A-buf 326 as disclosed above). In a preferred embodiment of the invention, the CLEAR signal and the STORE-ENABLE signal for the Z-mem 310 may be transmitted on separate lines from similar signals for the A-mem 314.

Each operation previously disclosed is performed by one or more PSRs 1510 operating with the bank 110, array 112, and mnemond 114 structure shown in Figure 3. Each of the store, locate, and retrieve operations has one or more PSRs 1510 which perform that operation. Several other primitive operations are provided to allow a memory system controller more flexibly to manipulate the memory system disclosed herein.

A "READ ADDRESS" operation is provided to read a memory address pointer 216 from an A-mem 314, when provided with a memory address pointer 216 to the A-mem 314 to be read from. The "K" signal on line 328a is triggered and the "Z" signal on line 328b is awaited, causing the A-CCR 328 to count its entire cycle, and thus causing A-bits 316 in the A-mem 314 to be retrieved. These A-bits 316 are ANDed using bank 110 AND gate 358 to retrieve a memory address pointer 216.

A "STORE ADDRESS" operation is provided to store a memory address pointer 216 into an A-mem 314, when provided with a memory address pointer 216 to the A-mem 314 to be stored into. The "K" signal on line 328a is triggered and the "Z" signal on line 328b is awaited, while the "STORE ENABLE" signal on line 352c is enabled, causing the A-CCR 328 to count its entire cycle, and thus causing A-bits 316 in the A-mem 314 to be altered to add a new memory address pointer 216.

A "RETRIEVE DATA STRUCTURE" operation is provided to retrieve information from one or more Z-mems 310, when provided with a memory address pointer 216 to the Z-mems 310 to be read from. Given a memory address pointer 216, the memory system controller determines which particular mnemonds 114 in the bank 110 store the beginning of the data structure 210 to be retrieved. Retrieval follows the steps disclosed with reference to Figure 5.

The "reset" signal on line 324c is triggered, which resets the Z-CCR 324, the A-CCR 328, and the match-CCR 332 simultaneously. The "B" signal on line 324a is then triggered, which starts the Z-CCR 324 counting for one Z-mem 310 cycle. The Z-CCR 324 cycles the Z-bits 312 in the Z-mem 310, causing them to be read out one by one for each mnemond 114 in the bank simultaneously. Each Z-bit 312 is ANDed with the corresponding AXN-bit 416 for that mnemond 114 by AND gate 356a, and only those Z-bits 312 for mnemonds whose AXN-bits 416 are set to a logic "1" are propagated to the output of AND gate 356. These Z-bits 312 are ANDed together by AND gate 356b to produce an output Q-bit 344, which is stored in the Q-buf 322 via mux 357 for readout by a memory system controller. The AXN-register 414 is then advanced one bit by triggering signal "R" on line 418a to shift the AXN-register 414 right.

The output data element 212 stored in the Q-buf 322 is also compared with an end of structure mark 214. If no end of structure mark 214 is detected, the operation for reading a data element 212 is triggered again, causing the next data element 212 to be retrieved from the next set of mnemonds 114. If an end of structure mark 214 is detected, the READ ADDRESS operation disclosed above is triggered, causing the memory address pointer 216 from the A-mem 314 then referenced by the AXN-register 414 to be retrieved and stored in the A-buf 326.

A "LOCATE CUED DATA STRUCTURE" operation is provided to locate a specific cue structure 410 in memory, when provided with a cue structure 410 to locate. Locating the cue structure 410 follows the steps

disclosed with reference to Figure 4. First, the "H" signal on line 338b is triggered, which clears the MATCH register 338.

The "E" and "F" signals on lines 330b and 318c respectively are then triggered, simultaneously clearing the match-ctr 318 and Q-ctr 330 registers to indicate zero cue data points. The "B" signal on line 324a is then triggered and the "X" signal on line 324b is awaited, causing the Z-CCR 324 to count its entire cycle, and thus causing the Z-mem 310 and Q-buf 322 to be compared. Note that this causes the Q-ctr 330 to be updated to indicate the number of cue data points in the cue element 412 presented, and causes the match-ctr 318 to be updated to indicate the number of cue data points which were matched. The "A" signal on line 332a is then triggered and the "W" signal on line 332b is awaited, causing the match-CCR 332 to count its entire cycle, and thus causing the match-ctr 318 and Q-ctr 330 to be compared. This completes the comparison of a single cue element 412.

The "R" signal on line 418a is then triggered to advance the AXN-register 414 one bit, followed by triggering the "P" signal on line 342a to enable the AXN-latch 342 and the "Q" signal on line 418b to reload the AXN-register 414 with the modified AXN-bits 416 (i.e. the AXN-bits 416 have been processed by gates 346a-c to yield only those AXN-bits 416 which continue to be associated with mnemonds 114 which have detected a match with the latest presented cue element 412). If there are no more cue elements 412 at this point the cueing operation is complete, and no more steps are taken. The located data structure 210 may then be retrieved with the "RETRIEVE DATA STRUCTURE" operation, disclosed above. If there are more cue elements 412, the cue operation is triggered again, causing the next cue element 412 to be presented to the next set of mnemonds 114.

A "STORE DATA STRUCTURE" operation is provided to store a specific data structure 210 in memory. Storing the data structure 210 follows the steps disclosed with reference to Figure 2. First, as disclosed with reference to Figure 5, the data strucuture 210 to be stored in memory is first presented to the memory as a cue structure 410, complete with start of structure mark 218 and an end of structure mark 214, to determine if that data structure 210 has already been stored somewhere in memory. If that data structure 210 is detected as already present in memory, no store operation is performed.

As disclosed with reference to Figure 2, a memory system controller must first select a single mnemond 114 in each array 112 into which the first data element 212a of the data structure 210 is to be stored. After this sub-operation is complete, the AXN-register 414 for each array 112 will comprise a set of AXN-bits 416 which are set to a logic "1" to indicate those mnemonds 114 where the first data element 212a is to be stored, and exactly one AXN-bit 416 will be set per array 112.

A single data element 212 is then stored. The "B" signal on line 324a is triggered and the "X" signal on line 324b is awaited, while the "STORE ENABLE" signal on line 352c is enabled, causing the Z-CCR 324 to count its entire cycle, and thus causing Z-bits 312 in the Z-mem 310 to be altered to add a new data element 212. If there are more data elements 212 to be stored, the "R" signal on line 418a is triggered to shift the AXN-register 414 one bit, and the store operation is triggered again, causing the next data element 212 to be stored.

If there are no more data elements 212 to be stored, a memory address pointer 216 is stored, in a preferred embodiment, in each A-mem 314 in each array 112 then pointed to by the AXN-register 414. As disclosed with reference to Figure 5, this memory address pointer 216 points back to the starting location of the data structure 210. The "K" signal on line 328a is triggered and the "Z" signal on line 328b is awaited, while the "STORE ENABLE" signal on line 325c is enabled, causing the A-CCR 328 to count its entire cycle, and thus causing A-bits 316 in the A-mem 314 to be altered to add a new memory address pointer 216. The storage operation is then complete, and no more steps are taken.

Figure 10 is a block diagram of a control system 1010 for use with a system of banks 110. At least one bank 110, comprising arrays 112 and mnemonds 114 as disclosed with reference to Figure 1, is operated under the control of a processing element 1012, assisted by sensor equipment 1014, buffers and interface registers 1016, PSR support hardware 1018, and address manipulation hardware 1020.

The processing element 1012 may comprise a commonly available general purpose processor, e.g. a microprocessor chip, operating with working memory and operating under software control, as is well-known in the art. It will be clear to one of ordinary skill in the art that method steps disclosed herein may be generally implemented by one or more PSRs 1510 in the PSR support hardware 1018 as disclosed with reference to Figure 15, or by the processing element 1012 operating under software control, as is well-known in the art. For any particular set of method steps, it would be clear to one of ordinary skill in the art how to modify a PSR 1510 (disclosed with reference to Figure 15) to perform those method steps, and it would also be clear to one of ordinary skill in the art how to use commonly available processing elements to carry out those method steps under software control.

For example, the sensor equipment 1014 may comprise visual and/or voice sensory equipment for detecting visual and/or voice information in the ambient environment and encoding that visual and/or voice information for processing and storage in the memory system of the present invention. Other and further information on a sensor system which is particularly suited for use with the present may be found in European Patent Application No. (Attorney reference 31598/000) mentioned above.

The buffers and interface registers 1016 may comprise an address register 1022 for storing memory address pointers 216 and a data register 1024 for storing data structures 210. The address register 1022 is used to store values into and retrieve 110, translated between multiple representations, and manipulated to trace and recover the associative relations of data structures 210 which are stored in the bank 110.

Address values (in address symbol format) may be generated by an address generator 1112. It records the most recently generated address symbol in a current address register 1114 and generates the "next" address symbol upon command from the processing element 1012. Address symbols are further disclosed with reference to Figure 2.

Address values (in memory address pointer 216 format) may be retrieved from their stored representation in the bank 110 by an address demultiplexer 1116. It retrieves the representations of address values stored in a designated A-mem 314 of each array 112 of the bank 110, and extracts address values by comparison of the differences between these representations. The address demultiplexer 1116 is further disclosed with reference to Figure 12.

Address values are translated between multiple representations by an X/Y-to-Z address or data translator 1118 and a Z-to-X/Y address or data translator 1120. The X/Y-to-Z address or data translator 1118 converts address values from a (two dimensional) "X/Y" format, disclosed with reference to Figure 8, to a (one dimensional) "Z" format, which is identical to the format in which data items 212 are stored in a Z-mem 310. The reverse operation is performed by the Z-to-X/Y address or data translator 1120. The X/Y-to-Z address or data translator 1118 and the Z-to-X/Y address or data translator 1120 are further disclosed with reference to Figures 13A and 13B respectively.

Address values (in Z-format) may be manipulated to trace and recover the associative relations of data structures 210 which are stored in the bank 110 by an address list associator 1122. The address list associator 1122, using an address value which has been retrieved from the bank 110 (i.e. retrieved by the address demultiplexer 1116 and translated by the X/Y-to-Z address or data translator 1118), traces all address values which are associated with the original address value by memory address pointers 216. Associated address values which are associated with the original address value by, e.g., links of a specified indirection depth, or of a specified quality level or abstraction level, may be specified, and these informational tags may be recovered. The address list associator 1122 is further disclosed with reference to Figure 14.

Figure 12A is a block diagram of an address extractor and demultiplexer 1116. The content of the A-mem 314 of the selected mnemond 114 for each array 112 is stored in a set of A-bufs 1210a-z (e.g. 1210i) and the memory address pointer 216 which is common to them is extracted. The address extractor and demultiplexer 1116 detects memory address pointers 216 which are common to all A-bufs 1210a-z, and extracts and records them for use by the control system 1010.

In a preferred embodiment, the address extractor and demultiplexer 1116 may operate by comparing each pair of A-bufs 1210a-z for common memory address pointers 216, using each possible bit offset so that memory address pointers 216 which are offset with respect to each other will still be detected. Each A-buf 1210a-z can then be examined for the presence of a detected common memory address pointer 216, so that only memory address pointers 216 which are present in all A-bufs 1210a-z are actually extracted. The address extractor and demultiplexer 1116 makes use of a set of A-bufs 1210 (e.g. 1210a-z), an A-buf CCR 1212, a first AXOND gate 1214, a first A-match 1216, a set of second AXOND gates 1218 (e.g. 1218c-z), and a set of second A-matches 1220 (e.g. 1220c-z).

Each A-buf 1210 (e.g. 1210i) records a copy of the contents of each selected A-mem in each array 112. The A-buf CCR 1212, a CCR 1610 twice the length of each A-buf 1210, is used to "slide" A-buf 1210b past A-buf 1210a (i.e. compare the two A-bufs 1210 with each possible bit offset). A first AXOND gate 1214 responds when there is a common pattern (three or more identical bits, in a preferred embodiment) in the two A-bufs 1210, emits a pulse signal on line 1222a when this condition is detected, and records the common pattern in a first A-match 1216.

The contents of the first A-match 1216 is copied into each second A-match 1220c-z (one per A-buf 1210c-z, excluding the first two A-bufs 1210a-b). The A-buf CCR 1212 then "slides" each A-buf 1210c-z past its corresponding A-match 1220c-z. A set of second AXOND gates 1218c-z responds when the common pattern appears in the A-buf 1210c-z and its corresponding A-match 1220c-z, and emits a pulse signal on line 1222c-z when this condition is detected.

The pulse signals on lines 1222a and 1222c-z are recorded in a set of corresponding latches 1224a and 1224c-z. The outputs of these latches 1224a and 1224c-z are electrically connected to the inputs of an AND gate 1226, which outputs a logic "1" on line 1228 if its inputs are all logic "1". This output on line 1228, along with the contents of the first A-match 1216, may be transmitted to the control system 1010 upon command.

Figure 12B is a block diagram of an AXOND gate 1214. Operation of the first AXOND gate 1214 and the set of second AXOND gates 1218c-z is identical. The AXOND gate 1214 responds when there are three or more (as determined by a threshold) identical bits in its two inputs, emits a pulse signal when this condition is detected, and records the common pattern in an output register 1278. For clarity, in Figure 12A, an AXOND gate 1214 is shown with its pulse signal as an output and with its output register as a separate register.

A set of input lines 1250 propagates the first input to the AXOND gates 1214. Each input is electrically connected to a pull-down resistor 1252 and to the parallel input of an input register 1254 comprising a set of bits 1256. In a preferred embodiment, the input register 1254 may comprise a parallel-serial shift register such as the 7496 parallel-serial shift register. The input register 1254 is loaded with the second input to the AXOND gate 1214. Each pull-down resistor 1252 prevents the corresponding bit 1256 of the the input register 1254 from being parallel-loaded with a logic "1" unless the first input bit on a line 1250 is also a logic "1"; thus this circuit performs a bitwise logical AND on the two inputs.

Each bit 1280 of the output register 1278 is also propagated to the input of a corresponding AND gate 1282,

along with a common CLEAR line 1284, and the output of each AND gate 1282 is propagated to an input of the corresponding OR gate 1286. Thus, the output register 1278 will record the pattern present in the input register 1256 if and only if the detection pulse on line 1270 is triggered. If the CLEAR line 1284 is raised, the output register 1278 will be cleared before a new pattern is recorded; otherwise the new pattern will overlay the old pattern.

Figure 13A is a block diagram of an X/Y-to-Z address or data translator 1118. Memory address pointers 216 are recorded in an X/Y format (disclosed with reference to Figure 8), a two-dimensional format comprising an X-part and a Y-part and representing an address value equal to (k1 * X) + (k2 * Y). In a preferred embodiment, k1 equals 1 and k2 equals the maximum allowed value of X. The X/Y-to-Z address or data translator 1118 converts an input address value from the X/Y format to a Z format (equivalent to the format in which data is stored in a Z-mem 310), a one-dimensional format in which a bit represents an address value equal to its bit offset.

In a preferred embodiment, the X/Y-to-Z address or data translator 1118 may operate by selecting one of a plurality of Z-format segments 1310a-z (e.g. 1310i) based on the value of the X-part, and advancing a set bit to a location on the selected Z-format segment 1310i based on the value of the Y-part. This causes the value (k1 * X) + (k2 * Y) to be calculated, as is well-known in the art. The X/Y-to-Z address or data translator 1118 makes use of a set of Z-format segments 1310a-z, an X/Yi-CCR 1312, an X/Yi-buf 1314, a Z-format CCR 1316, a Z-select CCR 1318, a set of X-part segments 1320a-z (e.g. 1320i) each associated with a Z-format segment 1310, an X-part CCR 1322, and a PULSE register 1324.

Initially, signal "B" on line 1326 is triggered to reset the Z-select CCR 1318 and the X-part CCR 1322. Signal "D" on line 1328 is then triggered to cause the X/Yi-CCR 1312 to count its entire cycle, and thus causing the X/Yi-buf 1314 to input a memory address pointer 216 in an X/Y format as serial input bits on line 1328.

The first indicator bit 802 of the memory address pointer 216 is propagated through AND gate 1330 to the PULSE register 1324 to trigger signal "R" on line 1332. When signal "R" is received, signal "F" on line 1334 is triggered to cause the Z-select CCR 1318 to count its cycle, thus alternately selecting each Z-segment 1310a-z and each associated X-part segment 1320a-z one per count.

The second indicator bit 802 of the memory address pointer 216 is also propagated through AND gate 1330 to the PULSE register 1324 to trigger signal "S" on line 1336. When signal "S" is received, cycling of the Z-select CCR 1318 is terminated, thus selecting a particular Z-segment 1310a-z (e.g. 1310i). Signal "G" on line 1338 is then triggered to reset each X-part segment 1320a-z, followed by signal "H" to cause each X-part segment 1320a-z to count its cycle one per count.

The third indicator bit 802 of the memory address poiner 216 is also propagated through AND gate 1330 to the PULSE register 1324 to trigger signal "T" on line 1340. When signal "T" is received, cycling of each X-part segment 1320a-z is terminated, thus selecting a particular bit within the selected Z-segment 1310a-z (e.g. 1310i).

Signal "V" on line 1342 is then awaited, indicating that the X/Yi-CCR 1312 has completed its cycle. Signal "J" on line 1344 is triggered to enable output from the Z-format segments 1310a-z. Signal "W" on line 1346 is then triggered, and the "X" signal on line 1348 is awaited, causing the X-part CCR 1322 to count its entire cycle, and thus causing the contents of each X-part segment 1320a-z to be transmitted to its corresponding Z-format segment 1310a-z.

After each memory address pointer 216 is input and translated, a "MORE" signal (not shown), which indicates that the control system 1010 has further memory address pointers 216 to be translated, is checked. If there are further memory address pointers 216 to be translated, control is transferred to the beginning (just after the reset operation using signal "B") to form a loop. Else, if no further memory address pointers 216 are to be translated, control continues as follows and the presently collected set is output.

To output the collected set of address values, signal "L" on line 1350 is triggered to enable output from the set of Z-format segments 1310a-z. Signal "Y" on line 1352 is then triggered, and the "Z" signal on line 1354 is awaited, causing the Z-format CCR 1316 to count its entire cycle, and thus causing the Z-format address value to be output on line 1356.

Figure 13B is a block diagram of a Z-to-X/Y address or data translator 1120. As disclosed with reference to Figure 13A, memory address pointers 216 are recorded in an X/Y format, but it is often desirable to manipulate address values in a Z format instead. The Z-to-X/Y address or data translator 1120 converts an input address value from the Z format to the X/Y format. In a preferred embodiment, the Z-to-X/Y address or data translator 1120 may operate by dividing Z-format pulses by a predetermined divisor to create an X-part and a Y-part. The Z-to-X/Y address or data translator 1120 makes use of an Xo-buf 1350, an Xo-CCR 1352, a Yo-buf 1354, a Yo-CCR 1356, an X/Yo-buf 1358, an X/Yo-CCR 1360, a Zi-buf 1362, and a Zi-CCR 1364.

Initially, signal "E" on line 1366 is triggered to clear the Xo-buf 1350, the Xo-CCR 1352, the Yo-buf 1354, the YoCCR 1356, and the X/Yo-CCR 1360. Then signal "A" is triggered to cause the Zi-CCR 1364 to count its entire cycle, and thus causing the Zi-buf 1362 to input a Z-format address value as serial input bits on line 1368. Signal "H" on line 1370 is triggered to cause the Xo-buf 1350 to count its cycle, and thus causing a bit in that register to be shifted once per count of the Zi-CCR 1364. Signal "I" on line 1372 is triggered to allow recycling of bit values in the Xo-buf 1350. Signal "L" on line 1374 is triggered to enable clocking of the Yo-buf 1354. Signal "J" on line 1376 is triggered by a complete cycle of the Xo-buf 1350, and itself triggers a single clock cycle of the Yo-buf 1354.

Signals "H" on line 1370, "I" on line 1372, and "L" on line 1374 are then re-triggered, and signal "R" on line

1378 is triggered to reset the X/Yo-buf 1358. Signal "D" on line 1380 is awaited to indicate that the entire Z-format address value has been processed. At this point, an X-part and a Y-part have been generated and must be assembled into an X/Y-format memory address pointer 216. Signal "N" on line 1382 is triggered to cause the X/Yo-CCR 1360 to count its cycle, and thus causing the X/Yo-buf 1358 to enter data; signal "P" on line 1384 is triggered to enable the X/Yo-buf 1358 to be clocked. Signal "V" on line 1386 is triggered to enable the Yo-buf 1354 to recycle its data. Signal "Y" on line 1388 is triggered to cause the Yo-CCR 1356 to count its cycle, and thus causing the Yo-buf 1354 to output its data as signal "K" on line 1390, which is entered into the X/Yo-buf 1358 through OR gate 1392.

Signals "V" on line 1386 and "P" on line 1384 are then re-triggered, and signal "K" on line 1390 is awaited. At this point, the Y-part has been assembled into the X/Y-format memory address pointer 216 in the X/Yo-buf 1358. Signals "P" on line 1384, "I" on line 1372 are re-triggered, and signal "X" on line 1394 is triggered to cause the Xo-CCR 1352 to count its cycle, and thus causing the Xo-buf 1350 to output its data as signal "J" on line 1376, which is entered into the X/Yo-buf 1358 through OR gate 1392. Finally, signal "P" on line 1384 is re-triggered, thus causing the X/Yo-buf 1358 to output the X/Y-format memory address pointer 216, and signal "U" on line 1396 is awaited to indicate that the X/Yo-CCR 1360 has completed its cycle. If further Z-format address values are presented by the control system 1010, the Z-to-X/Y address or data translator 1120 continues with its initial step to form a loop.

Figure 14 is a flow chart of a method followed by an address list associator. The address list associator 1122 compiles a list of address values which are associated with the original address value. It performs its functions as shown in the flow chart. The X/Y-to-Z address or data translator 1118 performs the actual compiling of a list of address values, as disclosed with reference to Figure 13A.

At step 1450, the address list associator 1122 stores the original address values in a first address list register 1410. At step 1452, the address list associator 1122 retrieves all address values from the bank 110 which are associated with address values found in the first address list register 1410, and stores these associated address values in a second address list register 1412. At step 1454, the address list associator 1122 compares the address values in the second address list register 1412 with the address values in the first address list register 1410. If additional address values have been found, control is transferred to step 1452 to form a loop; else the address values found in the first address list register 1410 are the complete list, and no further steps are taken.

The value of this operation, in the context of parallel associative contents-addressable electronic memory systems, can be illustrated with an example. The control system 1010 may direct the input to the bank 110 of several cue structures 410, and may then direct association lists of these cue structures to be compared for common elements. Software to perform this comparison would be clear to one of ordinary skill in the art and is thus not further disclosed herein. Although the compilation of the list is performed by special hardware, software methods are considered adequate to perform the comparison. The result of this two step operation is recognition by the control system of common features of the cue structures 410 which were input.

Thus, data structures 210 for each President of the United States (e.g. "WASHINGTON", "ADAMS", "JEFFERSON") may be stored in the bank 110, along with memory address pointers 216 to associated data structures. Further, data structures 210 for each political party (e.g. "DEMOCRAT", "REPUBLICAN") may also be stored in the bank 110, along with memory address pointers 216 to associated data structures. Still further, data structures 210 comprising other remarks (e.g. "ASSASSINATED", "DIED-IN-OFFICE", "RE-ELECTED") may also be stored in the bank 110, along with memory address pointers 216 to associated data structures. Then, if the control system 1010 causes cue structures 410 for several Presidents of the United States to be input to the bank 110, address values for data structures 210 which name common features of the Presidents named may be extracted.

For example, the data structure 210 "LINCOLN" would be associated with the data structures 210 "REPUBLICAN", "ASSASSINATED", "DIED-IN-OFFICE", and "RE-ELECTED"; the data structure 210 "McKINLEY" would be associated with the data structures 210 "REPUBLICAN", "ASSASSINATED", and "DIED-IN-OFFICE"; and the data structure 210 "ROOSEVELT" would be associated with the data structures 210 "DEMOCRAT", "DIED-IN-OFFICE", and "RE-ELECTED"; this is shown in the table below.

LINCOLN	REPUBLICAN	ASSASSINATED	DIED-IN-OFFICE	RE-ELECTED
McKINLEY	REPUBLICAN	ASSASSINATED	DIED-IN-OFFICE	
ROOSEVELT	DEMOCRAT		DIED-IN-OFFICE	RE-ELECTED

By presenting the data structures 210 for these three Presidents, the control system 1010 can recover the facts that both Lincoln and McKinley were Republicans, that both were assassinated, that both Lincoln and Roosevelt were re-elected, and that all three Presidents died in office. By comparison of these results, the control system 1010 can also recover the fact that all Presidents who are assassinated must die in office. Similarly, by presenting the data structures for a set of qualities, the control system can recover a list of Presidents with those qualities.

Claims

1. A memory bank for storing at least one data structure, said memory bank including
array means for storing a plurality of copies of said data structure;
means for recovering each stored data structure from said memory bank by examination of differences between at least two stored representations of said data structure in said set of array means.

2. A memory bank as claimed in claim 1, including means for receiving a cue structure; and means for generating a signal when said cue structure is equal to one of said stored data structures.

3 A memory bank as claimed in claim 2, wherein said signal is for indicating a memory location.

4. A memory bank as claimed in claim 1, including exactly one said array means.

5. A memory bank for storing at least one data structure having a plurality of data elements, said memory bank including
at least one array means each for storing at least one copy of said data structure, each said array means itself including a plurality of mnemond means each for storing at least one of said data elements.

6. A memory bank as claimed in claim 5, each said array means including
means for selecting a set of said mnemond means;
means for presenting a cue element to said set of mnemond means; and
means for recording a response from each one of said set of mnemond means.

7. A memory bank as claimed in claim 6, each said array means including
means for generating a signal when said selected set of mnemond means consists of a predetermined number of mnemonds.

8. A memory bank as claimed in claim 7, wherein said predetermined number is one.

9. A memory bank as claimed in claim 6, including
means for generating a signal when said selected set of mnemond means in each array consists of a predetermined number of mnemonds.

10. A memory bank as claimed in claim 9, wherein said predetermined number is one.

11. A memory bank as claimed in claim 5, wherein each said array means includes a differing number of said mnemond means.

12. A memory bank as claimed in claim 5, including
means for receiving a cue structure having at least one cue element in a set having a predetermined order;
means for presenting said cue elements in said predetermined order to each said array means; and
means for generating a signal if said cue structure is equal to a stored data structure having said set of cue elements in the same order.

13. A memory bank as claimed in claim 12, wherein said plurality of mnemond means in each said array means has a predetermined order.

14. A memory bank for storing at least one data structure having a plurality of data elements, comprising at least one array means, each for storing at least one copy of said data structure, said array means itself comprising a plurality of mnemond means, each for storing at least one of said data elements; and
said memory bank, being responsive when presented with a cue structure, for generating a signal indicative of a memory location when said cue structure is equal to one of said data structures stored therein.

15. A memory bank for storing at least one data structure having a plurality of data elements, comprising at least one array means, each for storing at least one copy of said data structure, said array means itself comprising a plurality of mnemond means, each for storing at least one of said data elements;
each said array means being responsive when presented with a cue structure, for generating a first signal when said cue structure is equal to one of said data structures stored therein; and
said memory bank being responsive when presented with a cue element, for generating a second signal indicative of a memory location when each said array means generates said first signal.

16. A memory bank for storing at least one data structure having a plurality of data elements, comprising at least one array means, each for storing one copy of said data structure, said array means itself comprising a plurality of mnemond means, each for storing at least one of said data elements;
each said mnemond means being responsive when presented with a cue element, for generating a first signal indicative of a match when said cue element is equal to one of said data elements stored therein;
each said array means being responsive when presented with a cue structure, for generating a second signal when, for each said cue element of said cue structure, at least one said mnemond means generates said first signal; and
said memory bank being responsive when presented with a cue structure, for generating a third signal indicative of a memory location when each said array means generates said first signal.

17. A memory bank for storing at least one data structure having a plurality of data elements, comprising at least one array means, each for storing one copy of said data structure, said array means itself comprising a plurality of ordered mnemond means, each for storing at least one of said data elements;

each said mnemond means being responsive when presented with a cue element of an ordered cue structure, for generating a first signal indicative of a match when a first said cue element is equal to one of said data elements stored therein;
register means for storing each said first signal generated by each said mnemond means;
each said mnemond means also for generating said first signal when a non-first said cue element is equal to one of said data elements stored therein, and said register means records that said first signal was generated by a preceding mnemond associated with said mnemond when presented with a preceding cue element associated with said non-first cue element;
each said array means being responsive when presented with a cue structure, for generating a second signal when, for each said cue element of said cue structure, at least one said mnemond means generates said first signal; and
said memory bank being responsive when presented with a cue structure, for generating a third signal indicative of a memory location when each said array means generates said second signal.

18. A memory bank as claimed in claim 15,
said memory bank being responsive when presented with a cue structure, for generating a second signal indicative of a memory location when a predetermined number of said array means generates said first signal.

19. An array for storing at least one data structure having a plurality of data elements, comprising
a plurality of mnemond means, each for storing at least one of said data elements; and
said array means being responsive when presented with a cue structure, for generating a signal indicative of a memory location when said cue structure is equal to one of said data structures stored therein.

20. An array for storing at least one data structure having a plurality of data elements, comprising
a plurality of mnemond means, each for storing at least one of said data elements;
each said mnemond means being responsive when presented with a cue element of a cue structure, for generating a first signal indicative of a match when said cue element is equal to one of said data elements stored therein; and
said array means being responsive when presented with a cue structure, for generating a second signal indicative of a memory location when, for each said cue element of said cue structure, at least one said mnemond means generates said first signal.

21. An array for storing at least one data structure having a plurality of data elements, comprising
a plurality of ordered mnemond means, each for storing at least one of said data elements; each said mnemond means being responsive when presented with a cue element of an ordered cue structure, for generating a first signal indicative of a match when a first said cue element is equal to one of said data elements stored therein;
register means for storing each said first signal generated by each said mnemond means;
each said mnemond means also for generating said first signal when a non-first said cue element is equal to one of said data elements stored therein, and said register means records that said first signal was generated by a preceding mnemond associated with said mnemond when presented with a preceding cue element associated with said non-first cue element; and
said array means being responsive when presented with a cue structure, for generating a second signal indicative of a memory location when, for each said cue element of said cue structure, at least one said mnemond means generates said first signal.

22. A mnemond for storing at least one data element, including
a plurality of registers, each associated with a data value;
each register set to a predetermined state to indicate the presence of a data element with the data value associated with that register.

23. A mnemond as claimed in claim 22, wherein each said register is a single bit.

24. A mnemond as claimed in claim 22, including means for receiving a cue element; and
means for generating a signal when said cue element is equal to one of said stored data elements.

25. A data format for storing at least one data element, comprising
a plurality of bits, each for recording the presence of a specific data element;
each said bit being set to indicate the presence of said data element when said data element is stored therein.

26. A data format for storing a data structure having a plurality of ordered data elements, comprising
a plurality of ordered registers each for recording the presence of at least one data element;
each said data element associated with one said register recording the presence of said data element;
at least one said data element having a predecessor data element;
each said register associated with each said data element having a predecessor register associated with said predecessor data element associated with said data element.

27. A data format as claimed in claim 26, wherein said registers are arranged in a ring structure, whereby each said register has exactly one said predecessor register.

28. A data format for storing a data structure having a plurality of ordered data elements each with an original data element value and a counting offset from a first said data element, comprising
a plurality of ordered registers for recording the presence of at least one data element;
each said data element associated with one said register recording its presence;

at least one said data element having a predecessor data element;
each said register associated with each said data element having a predecessor register associated with said predecessor data element associated with said data element;
each said data element having its presence recorded in its associated register with a recorded data element value equal to a predetermined function of its said original data element value and its said counting offset.

29. A data format as claimed in claim 28, wherein said predetermined function is a linear combination of said original data element value and said counting offset.

30. A data format as claimed in claim 29, wherein said predetermined function is the sum of said original data element value and said counting offset, plus a predetermined constant.

31. A data format as claimed in claim 28, wherein a predetermined set of said original data values are associated with each said register.

32. A data format as claimed in claim 31, wherein said predetermined set of said original data values comprises the original data value for a "space" character.

33. A data format for storing at least one data element having a data element value, comprising:
a plurality of ordered bits, each having a bit number;
each data element indicated by the presence of a pair of said bits;
wherein said data element value of said data element equals the difference between said bit numbers of set said pair of said bits, plus a predetermined constant.

34. A data format for storing a plurality of ordered data elements, each having a data element value, comprising:
a plurality of ordered bits, each having a bit number; each data element indicated by the presence of a pair of set said bits;
wherein the said data element value of said element equals the difference between the bit number of the pair of set said bits, plus a predetermined constant;
said pair of set said bits comprising a low bit and a high bit, whereby said bit number of said high bit exceeds said bit number of said low bit;
and wherein for each said pair of ordered data elements, comprising a low data element and a high data element, whereby said low data element precedes said high data element, the sum of said bit numbers for said low bit and said high bit for said high data element exceeds the sum of said bit numbers for said low bit and said high bit for said low data element.

35. A data format as claimed in claim 34, wherein for each said pair of ordered data elements, said bit number for said low bit for said high data element exceeds said bit number for said low bit for said low data element.

36. A data format as claimed in claim 34, wherein for each said pair of ordered data elements, said bit number for said high bit for said high data element exceeds said bit number for said high bit for said low data element.

37. A data format as claimed in claim 34, wherein for each said pair of ordered data elements wherein said low data element immediately precedes said high data element, said low bit for said high data element equals said high bit for said low data element.

38. A program shift register, including a plurality of program state bits having a predetermined order;
an active program state bit set to a predetermined value to indicate a program state; and
means for advancing said active program state bit from a program state bit to a next program state bit.

39. A program shift register as claimed in claim 38, including
means for generating a signal to indicate that said program shift register has entered a predetermined state.

40. A program shift register as claimed in claim 38, including
means for receiving an external signal; and
means for synchronizing entry into a program state with said external signal.

41. A program shift register as claimed in claim 40, wherein said external signal is a clock pulse.

42. A program shift register as claimed in claim 38, including
means for delaying advancing said active program state bit until a predetermined logical condition is satisfied.

43. An AXOND gate, including
first input means for receiving a first set of bits;
second input means for receiving a second set of bits;
means for computing a predetermined logical function of said first and second sets of bits; and
summator means for generating a signal when said predetermined logical function of said first and second sets of bits has a predetermined number of bits set to a predetermined value.

44. An AXOND gate as claimed in claim 43, wherein said predetermined logical function is a bitwise logical AND function.

45. An AXOND gate as claimed in claim 43, wherein said predetermined number of bits is three.

46. A pattern detector circuit, including means for comparing at least two input bit patterns for a common bit pattern;
means for comparing said common bit pattern with a set of additional input bit patterns; and

means for generating a signal when said common bit pattern is found in a predetermined number of said additional input bit patterns.

47. A circuit for translating a data value from an X/Y data format having at least two data elements to a Z data format having a single data element, said circuit including

a set of Z-format means, each for recording a data element;

means, responsive to at least one of said data elements of said X/Y data format, for selecting one of said Z-format means;

means, responsive to at least another one of said data elements of said X/Y data format, for storing a data element in said selected Z-format means; and

means, responsive to at least one data element stored in a Z-format means, for generating a data value in said Z data format.

48. A circuit for translating a data value from a Z data format having a single data element to an X/Y data format having at least two data elements, said circuit including

means for receiving an input data value in a Z data format;

a set of X shift register means, each for storing a bit, for advancing said bit a number of positions in response to said input data value, and for generating a recycle signal when said bit is recycled to a beginning location of said X shift register;

Y shift register means for storing a bit, and for advancing said bit a number of positions in response to said recycle signal; and

means, responsive to at least one X shift register and to said Y shift register, for transmitting an output value in an X/Y data format.

FIG. 1.

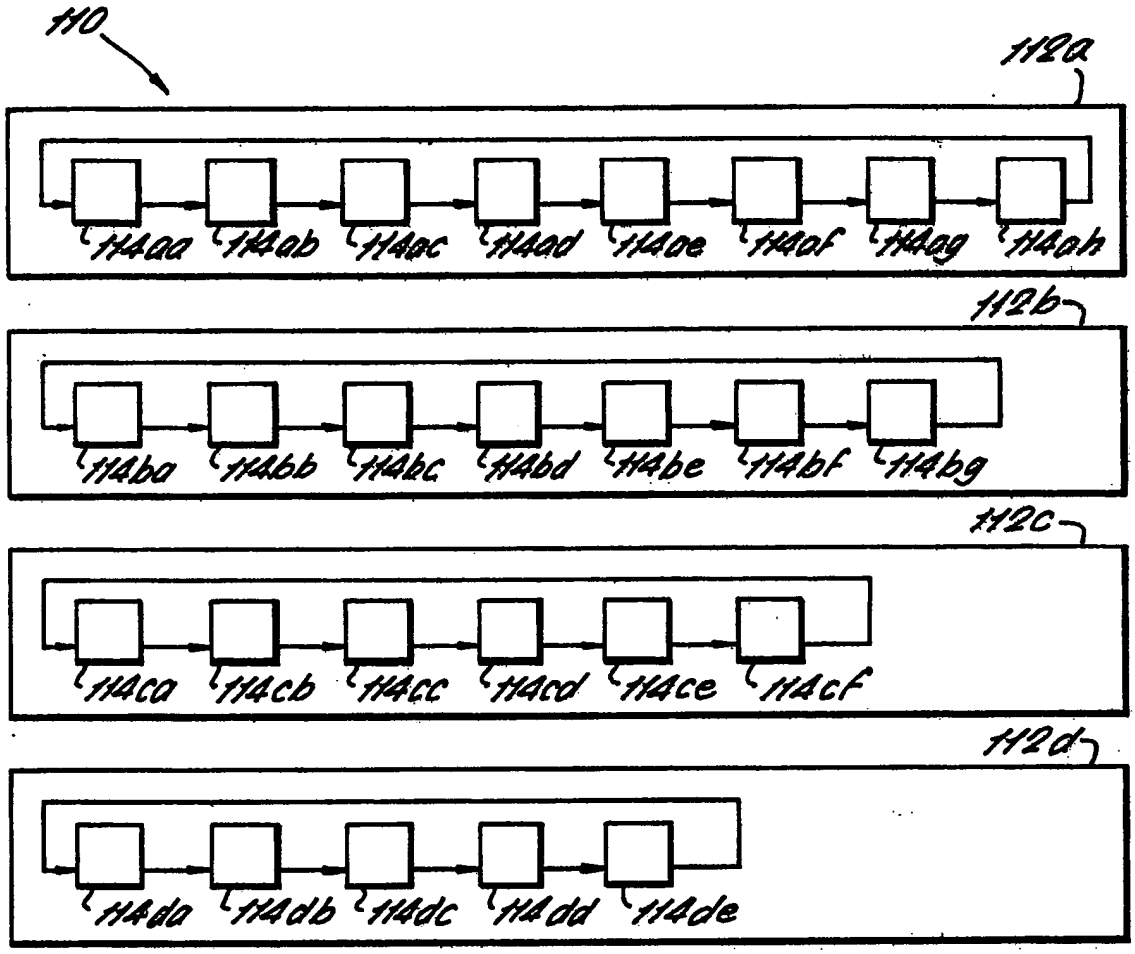

110
112a
114aa
114ab
114ac
114ad
114ae
114af
114ag
114ah
112b
114ba
114bb
114bc
114bd
114be
114bf
114bg
112c
114ca
114cb
114cc
114cd
114ce
114cf
112d
114da
114db
114dc
114dd
114de

FIG. 2.

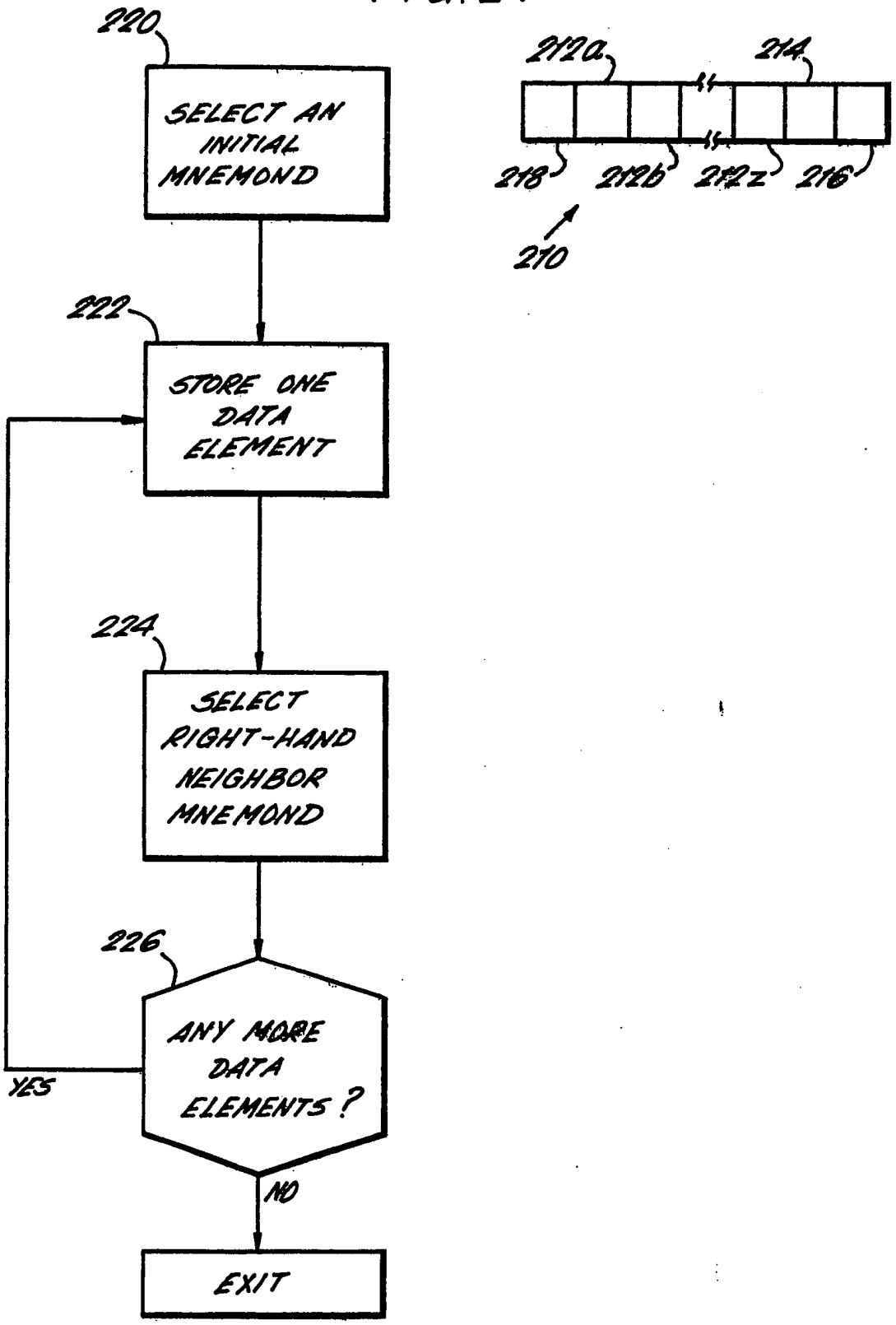

220
SELECT AN INITIAL MNEMOND
222
STORE ONE DATA ELEMENT
224
SELECT RIGHT-HAND NEIGHBOR MNEMOND
226
ANY MORE DATA ELEMENTS ?
YES
NO
EXIT
212a
214
218
212b
212z
216
210

FIG. 3.

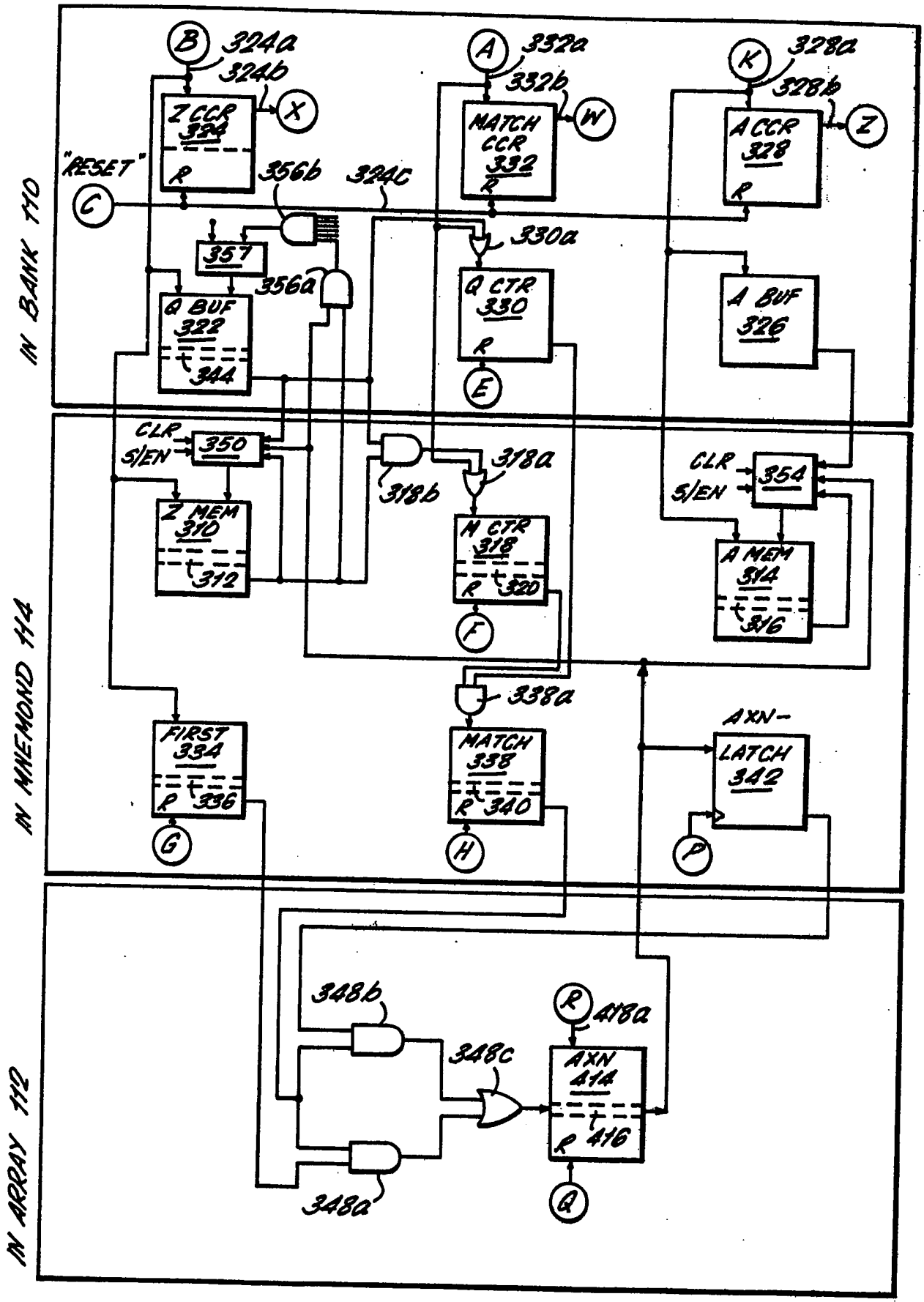

IN BANK 110
IN MNEMOND 114
IN ARRAY 112
B
324a
324b
Z CCR 324
R
X
"RESET"
C
356b
324c
357
356a
Q BUF 322
344
A
332a
332b
MATCH CCR 332
R
W
330a
Q CTR 330
R
E
K
328a
328b
A CCR 328
R
Z
A BUF 326
CLR S/EN
350
Z MEM 310
312
318b
318a
N CTR 318
R
320
F
CLR S/EN
354
A MEM 314
316
338a
FIRST 334
P 336
G
MATCH 338
R 340
H
AXN-LATCH 342
P
348b
348c
348a
R
418a
AXN 414
R 416
Q

FIG. 3A.

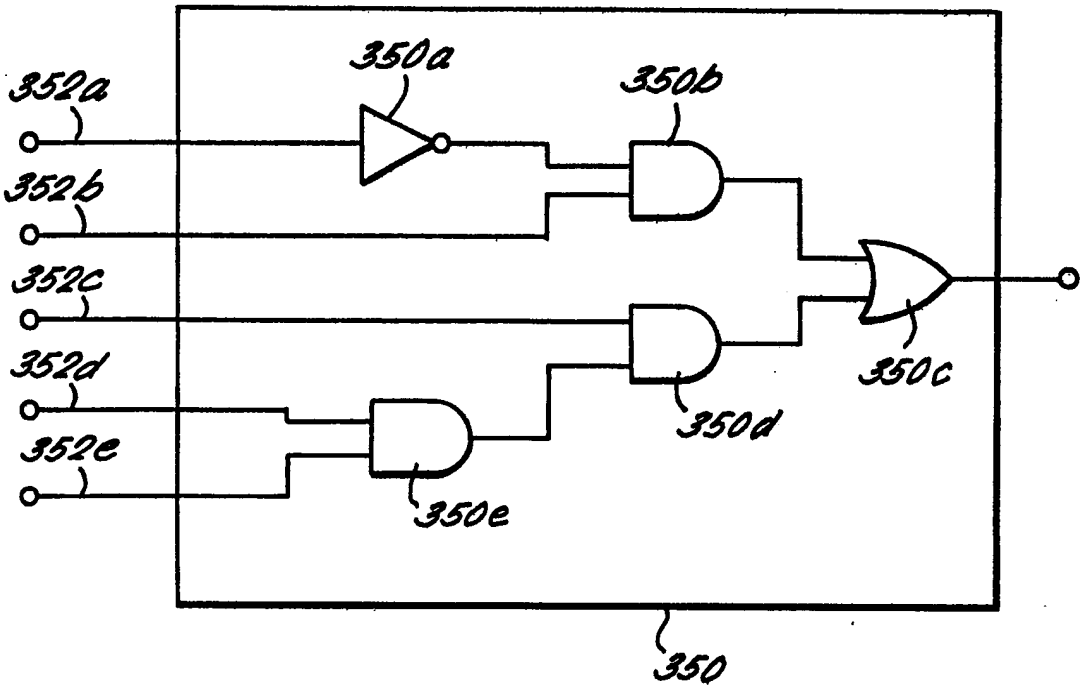
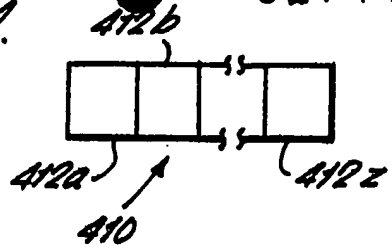


FIG. 4.


412b
412a
412z
410

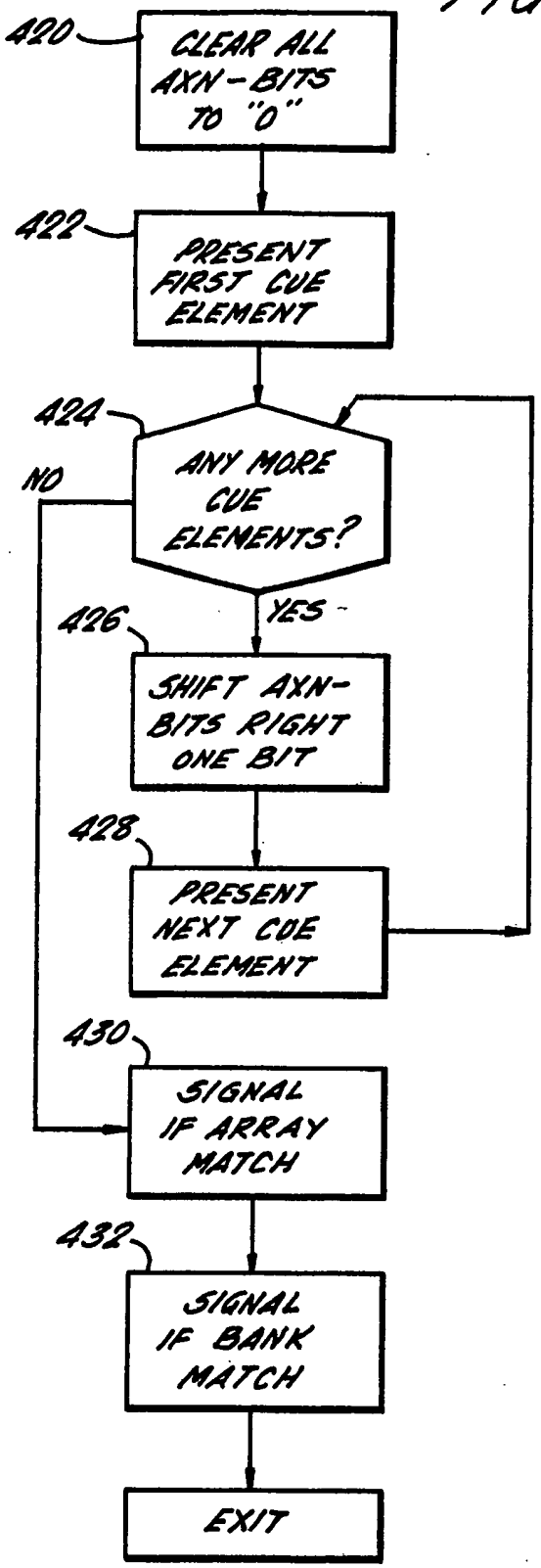

420
CLEAR ALL AXN-BITS TO "0"
422
PRESENT FIRST CUE ELEMENT
424
ANY MORE CUE ELEMENTS?
NO
YES
426
SHIFT AXN-BITS RIGHT ONE BIT
428
PRESENT NEXT COE ELEMENT
430
SIGNAL IF ARRAY MATCH
432
SIGNAL IF BANK MATCH
EXIT

FIG. 5.

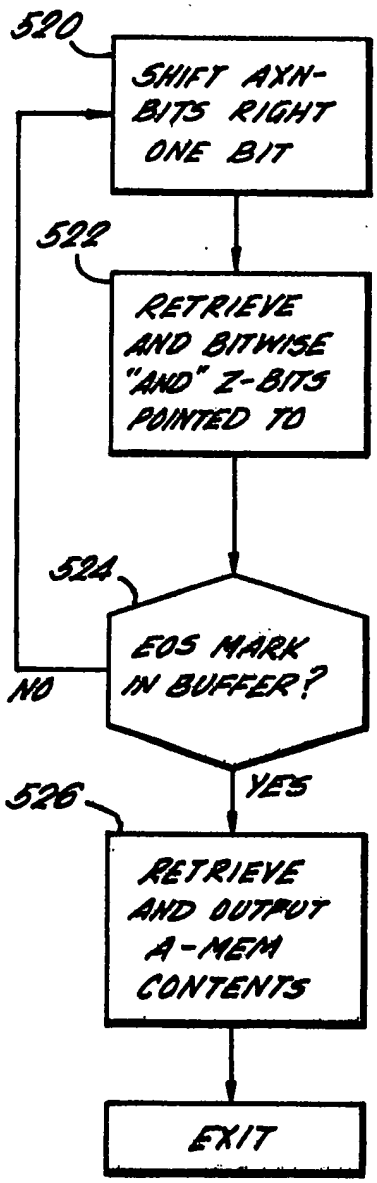

520
SHIFT AXN-BITS RIGHT ONE BIT
522
RETRIEVE AND BITWISE "AND" Z-BITS POINTED TO
524
EOS MARK IN BUFFER?
NO
YES
526
RETRIEVE AND OUTPUT A-MEM CONTENTS
EXIT

FIG. 6.

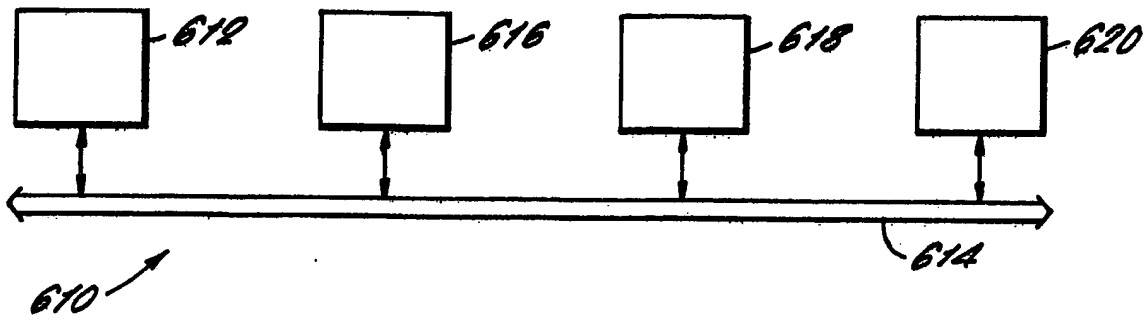


FIG. 7.

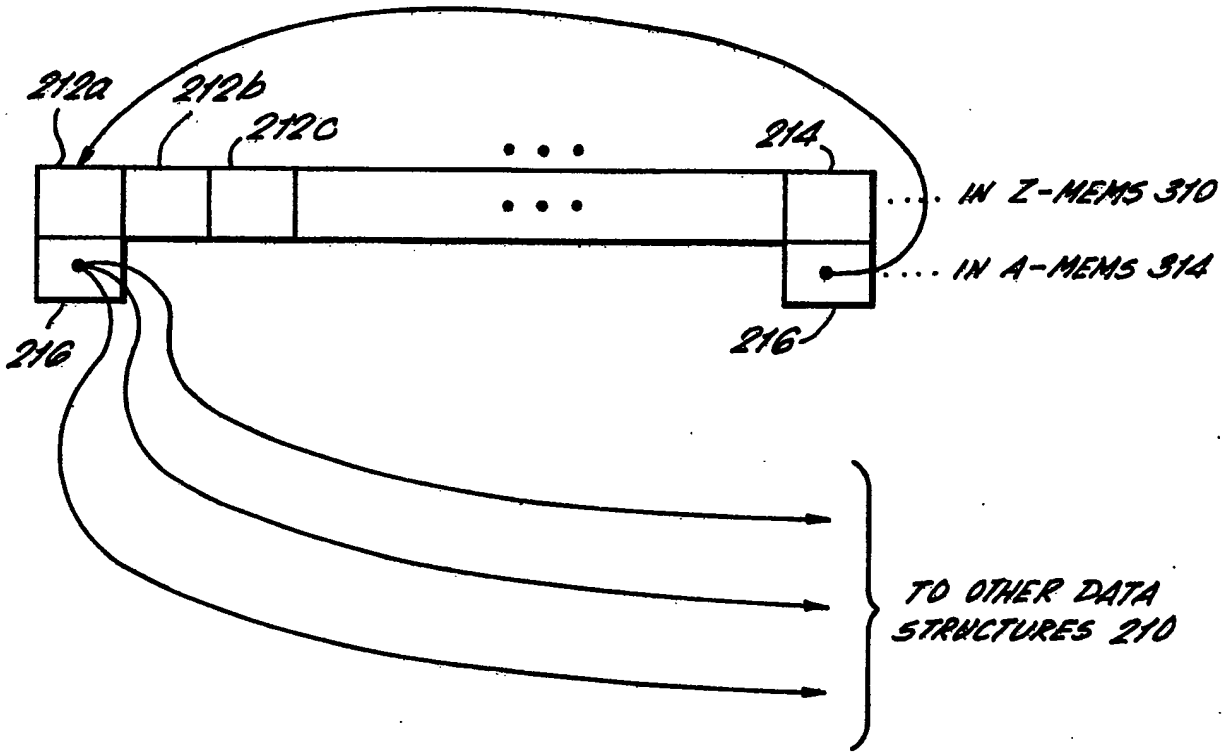


FIG. 8.

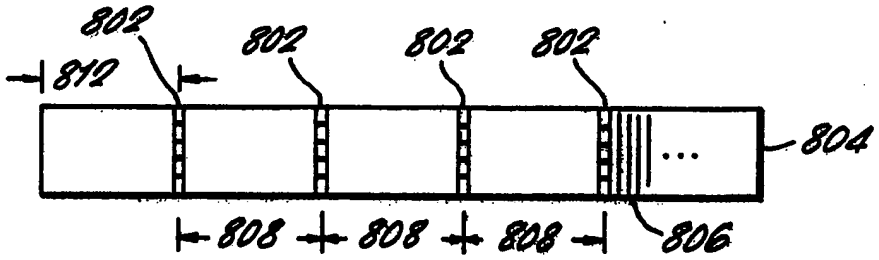


FIG. 9A.

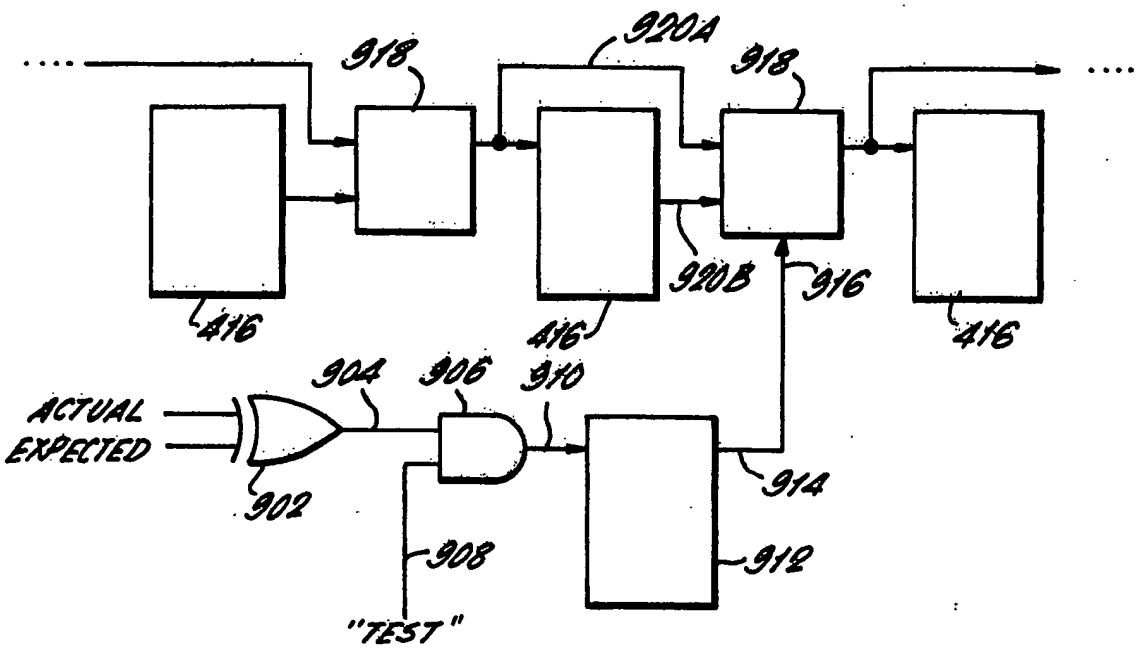

918
920A
918
920B
916
416
416
416
904
906
910
ACTUAL
EXPECTED
902
914
908
912
"TEST"

FIG. 9B.

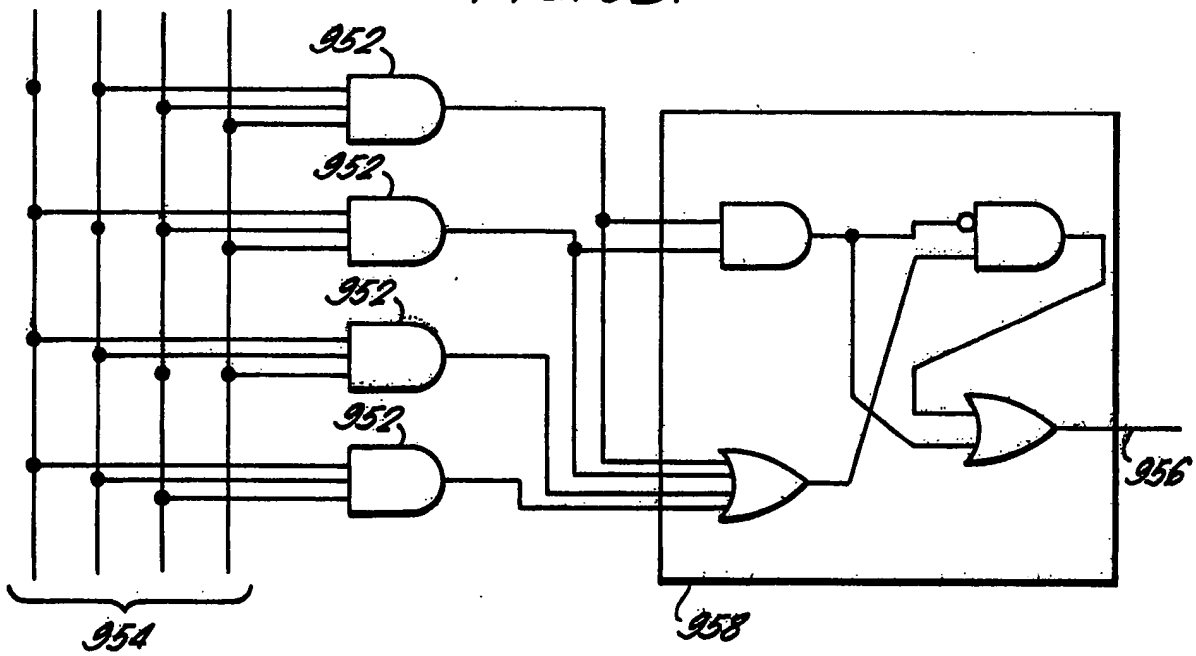

952
952
952
952
956
954
958

FIG. 10.

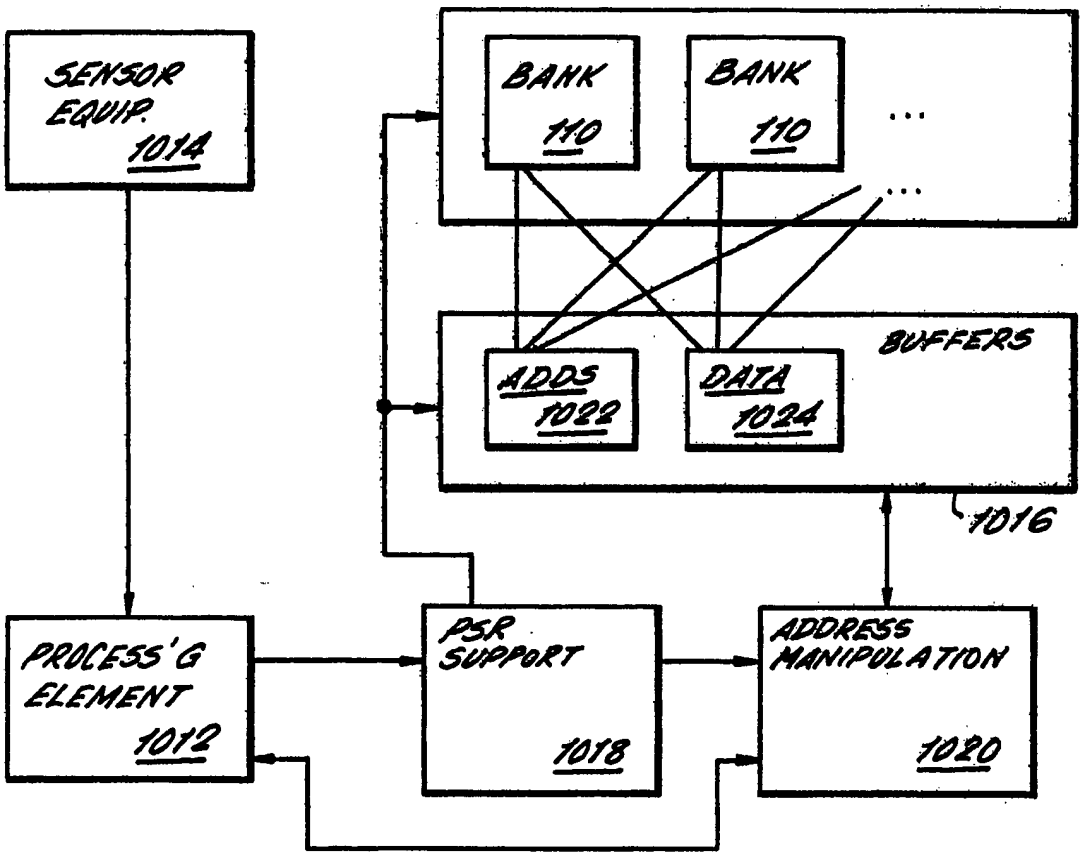

SENSOR EQUIP. 1014
BANK 110
BANK 110
BUFFERS
ADDS 1022
DATA 1024
1016
PROCESS'G ELEMENT 1012
PSR SUPPORT 1018
ADDRESS MANIPULATION 1020

FIG. 11.

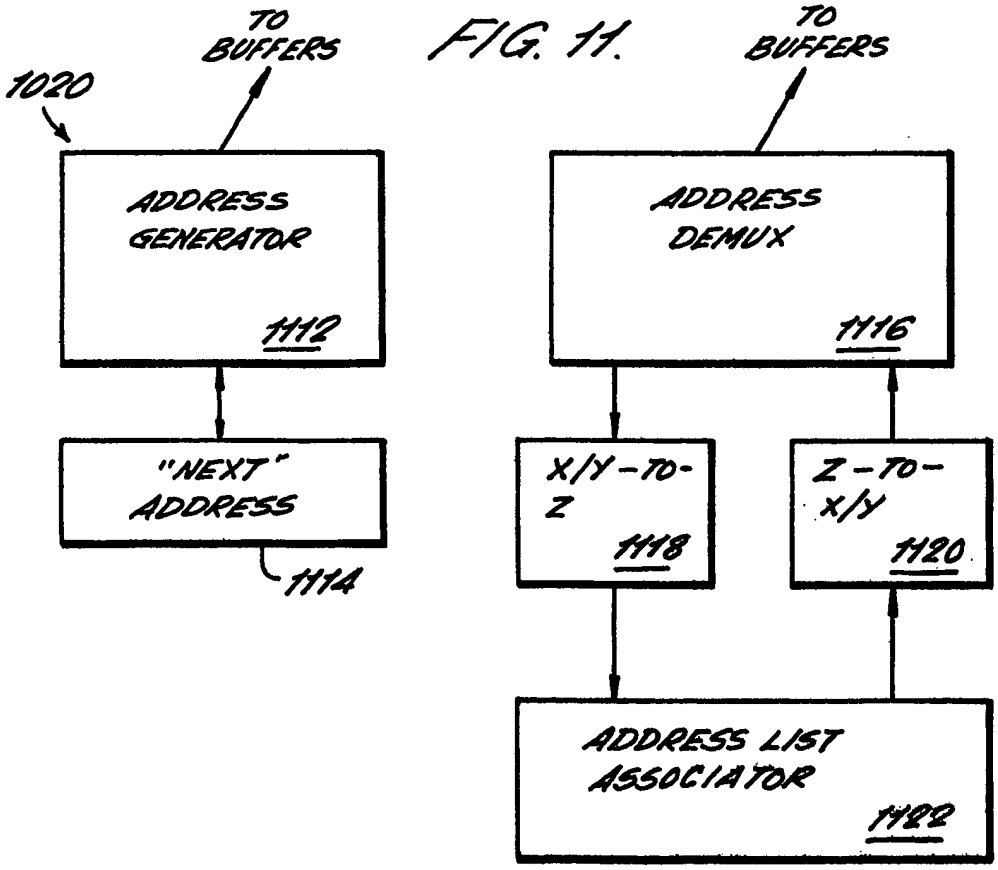

TO BUFFERS
1020
ADDRESS GENERATOR 1112
"NEXT" ADDRESS 1114
TO BUFFERS
ADDRESS DEMUX 1116
X/Y-TO-Z 1118
Z-TO-X/Y 1120
ADDRESS LIST ASSOCIATOR 1122

FIG. 12A.

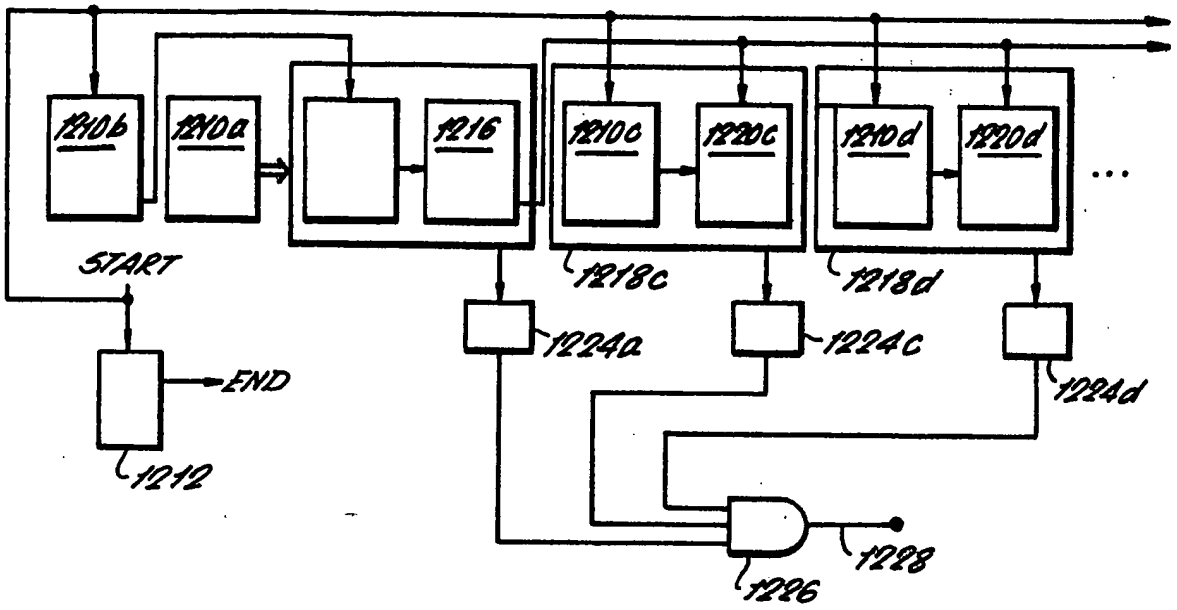


FIG. 12B.

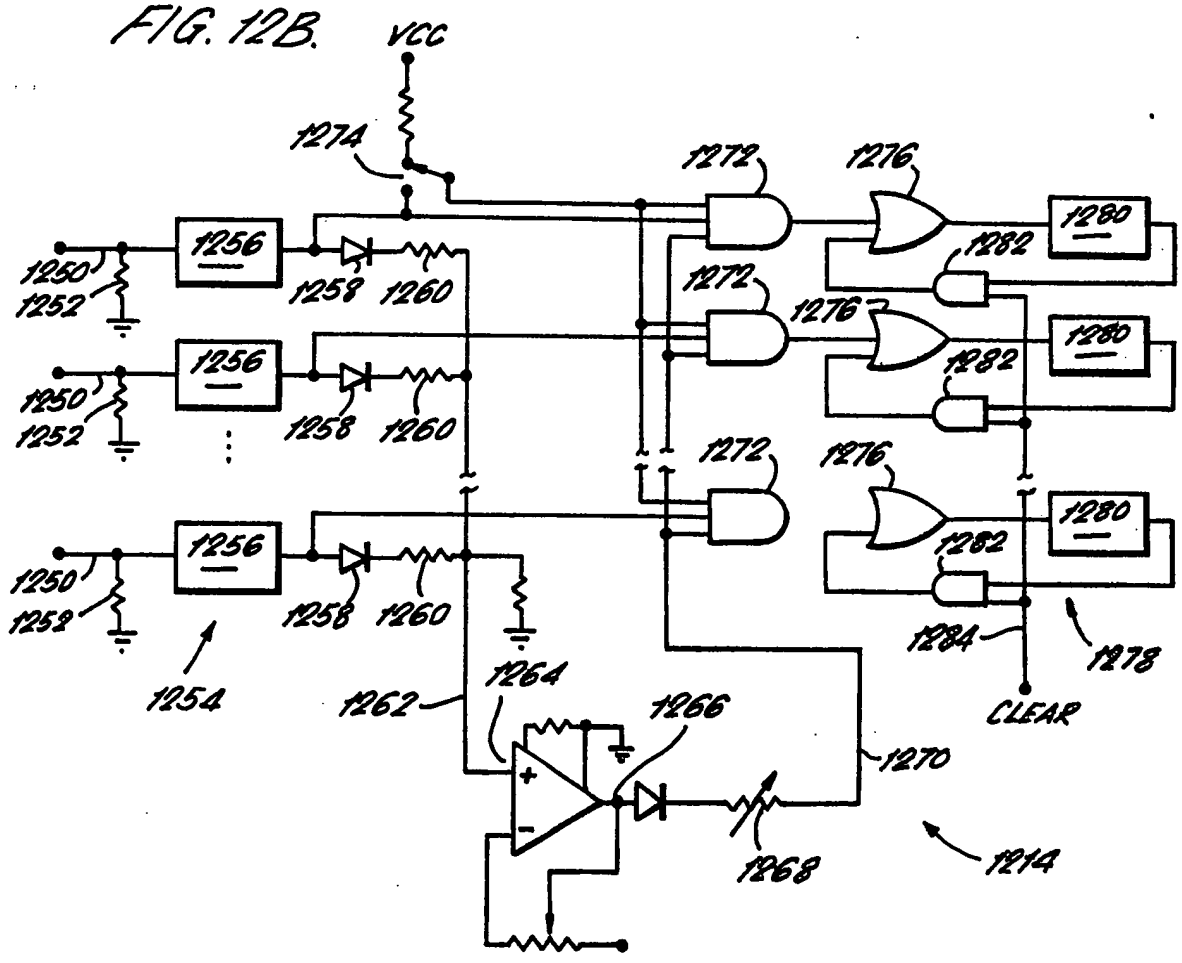


FIG. 13A.

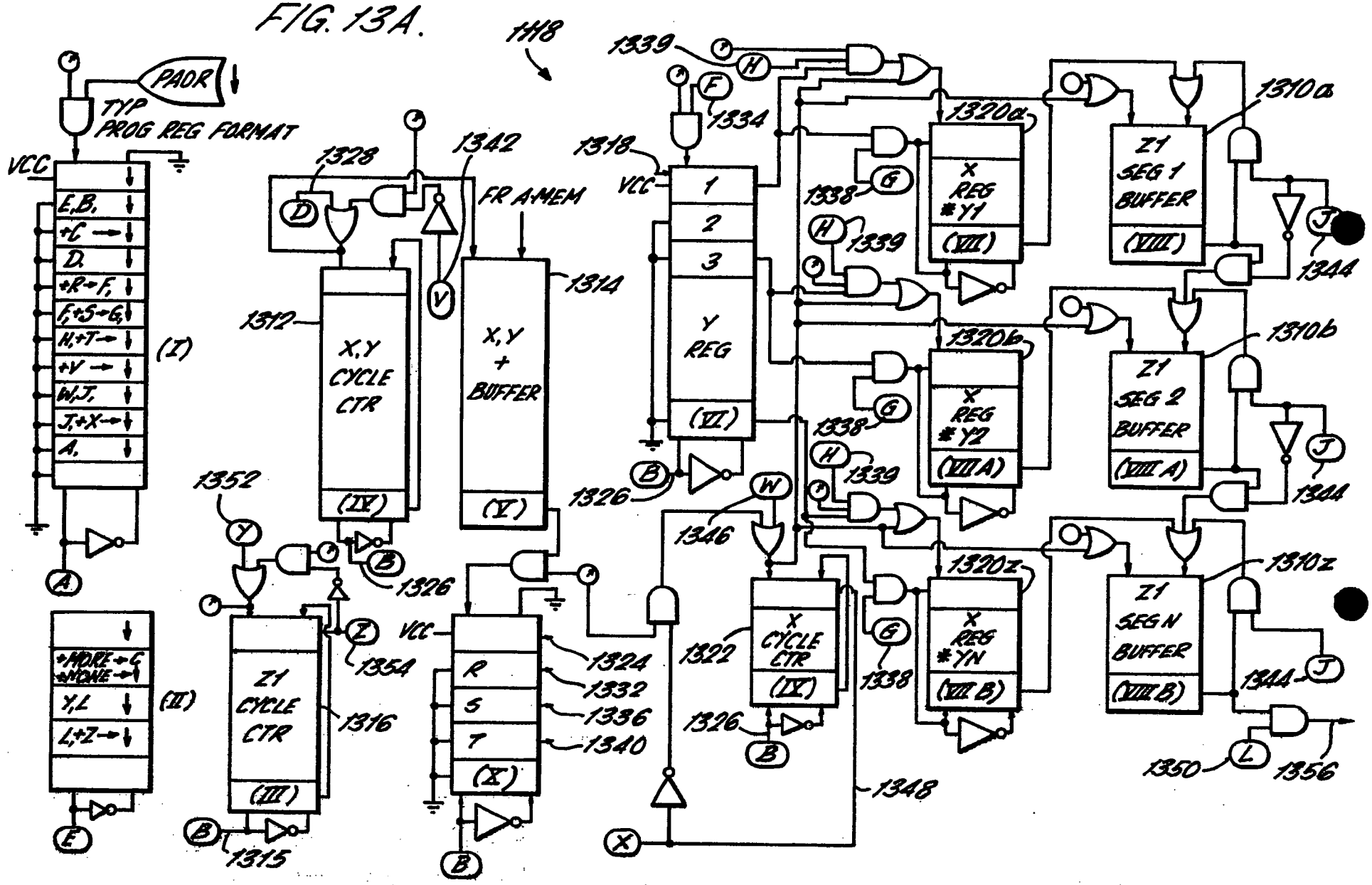
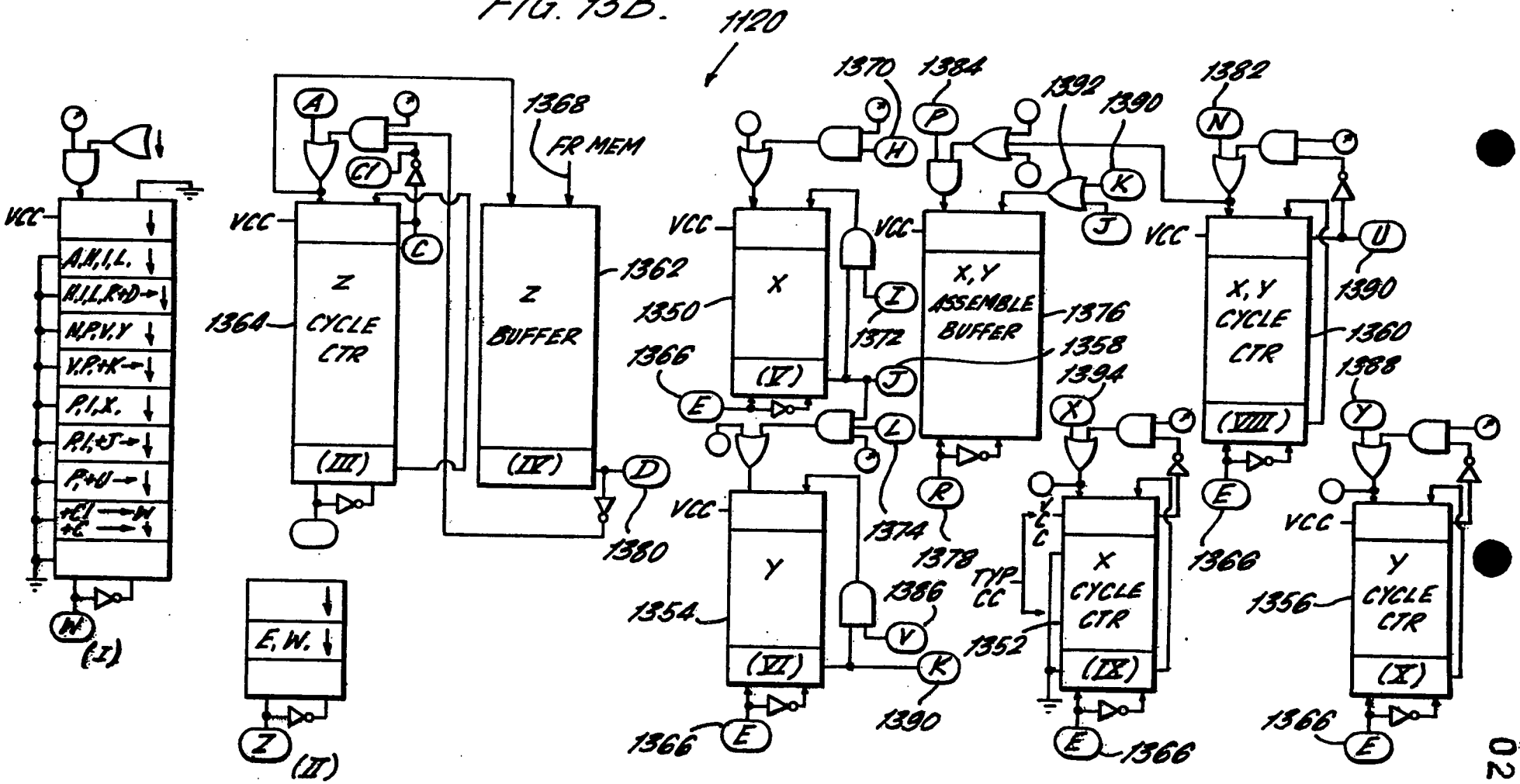

1318
PAOR
TYP PROG REG FORMAT
VCC
E,B,
+C
D,
+R→F,
F,+S→G,
H,+T→
+V
W,J,
J,+X→
A,
(I)
+MORE→C
+MORE→
Y,L
L,+Z→
(II)
1328
1342
FR A-MEM
D
V
1312
X,Y CYCLE CTR
(IV)
1314
X,Y + BUFFER
(V)
1352
Y
Z
1354
Z1 CYCLE CTR
1316
(III)
B
1315
1326
VCC
R
S
T
(X)
1324
1332
1336
1340
1339
H
F
1334
1318
VCC
1
2
3
Y REG
(VI)
1326
B
1338
G
1320a
X REG #Y1
(VIII)
1320b
X REG #Y2
(VIII A)
1320z
X REG #YN
(VIII B)
1346
W
1322
X CYCLE CTR
(IX)
1326
B
1348
X
1310a
Z1 SEG 1 BUFFER
(VIII)
1310b
Z1 SEG 2 BUFFER
(VIII A)
1310z
Z1 SEG N BUFFER
(VIII B)
J
1344
1350
L
1356

FIG. 13B.



0274426

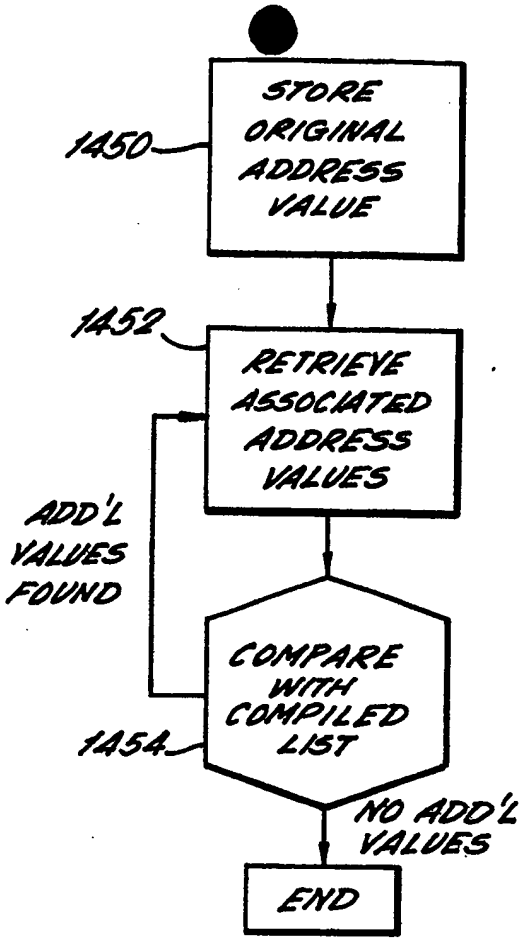

STORE ORIGINAL ADDRESS VALUE
1450
RETRIEVE ASSOCIATED ADDRESS VALUES
1452
ADD'L VALUES FOUND
COMPARE WITH COMPILED LIST
1454
NO ADD'L VALUES
END

FIG. 14.

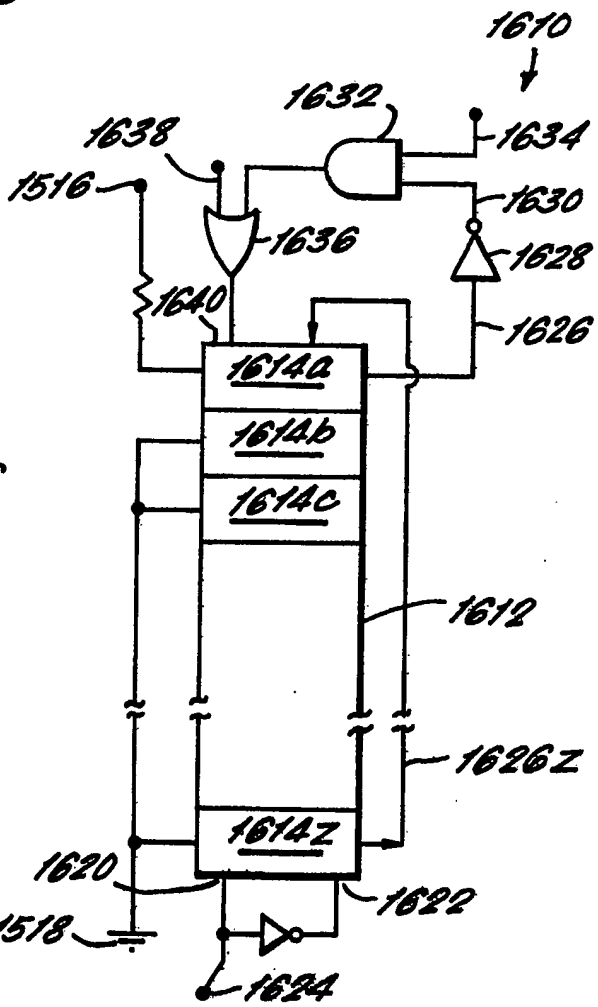
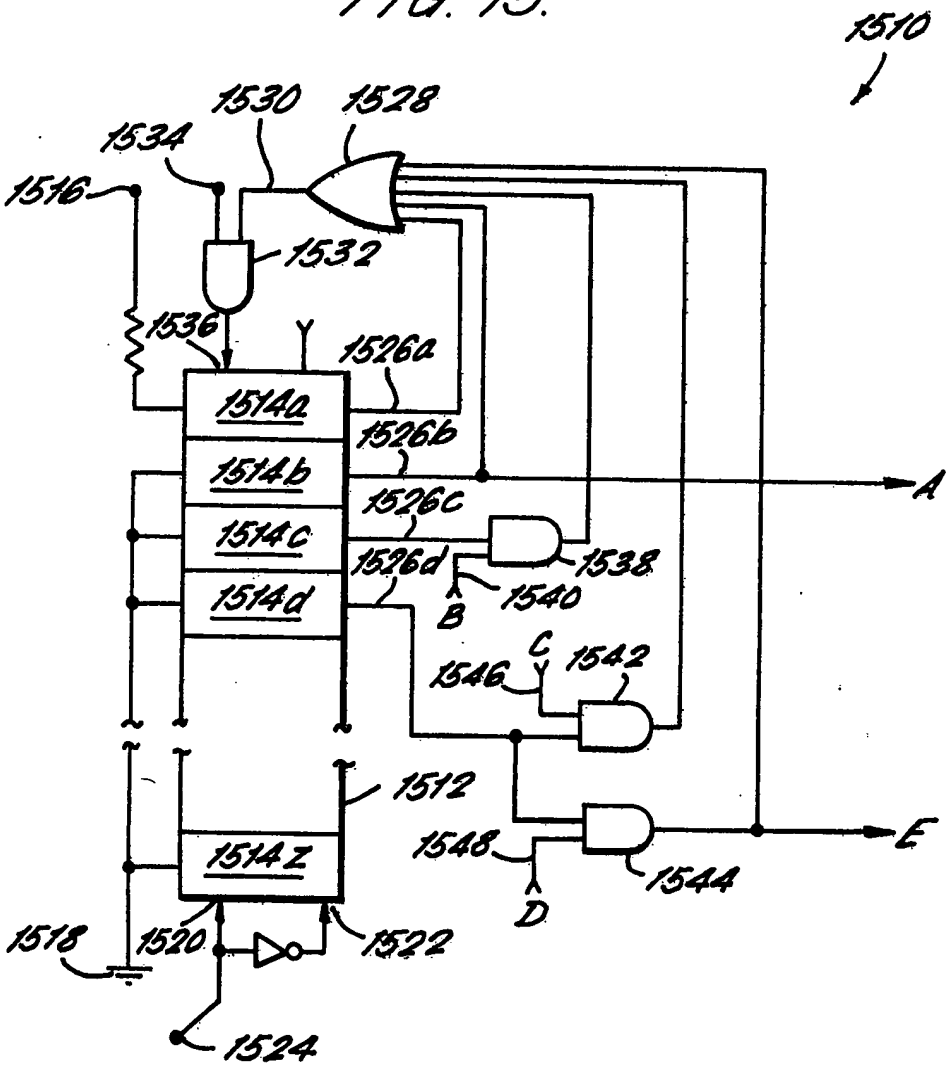

1610
1632
1634
1638
1516
1630
1636
1628
1640
1626
1614a
1614b
1614c
1612
1626z
1614z
1620
1622
1518
1624

FIG. 16.

FIG. 15.


1510
1530
1528
1534
1516
1532
1536
Y
1526a
1514a
1526b
1514b
A
1526C
1514C
1538
1526d
1540
1514d
B
C
1542
1546
1512
1514Z
1548
E
1544
D
1518
1520
1522
1524

THIS PAGE BLANK (USPTO)

THIS PAGE BLANK (USPTO)

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 274 426 A3**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **88300074.7**

(51) Int. Cl.5: **G06F 15/40**

(22) Date of filing: **06.01.88**

(30) Priority: **07.01.87 US 1233**

(43) Date of publication of application:
13.07.88 Bulletin 88/28

(84) Designated Contracting States:
DE FR GB IT

(88) Date of deferred publication of the search report:
21.10.92 Bulletin 92/43

(71) Applicant: **Nikken Foods Honsha Co., Ltd.**
Nikken Bldg. 305-8 1-Ban, 5-Chome,
Nishinakajima Yodogawa-Ku
Osaka(JP)

(72) Inventor: **Strehler, Bernard**
2235 25th Street
217 San Pedro California 90732(US)

(74) Representative: **Cross, Rupert Edward Blount et al**
BOULT, WADE & TENNANT 27 Furnival Street
London EC4A 1PQ(GB)

(54) Computer memory system.

(57) A plurality of memory units (114), called "mnemonds", is each capable of storing data elements and of responding to requests to recover those data elements from storage. A plurality of mnemonds is arranged in a linear array (112), with each mnemond connected to be capable of responding to signals from its neighbor mnemonds. When a data structure is stored, each array stores a copy of that data structure, either allocating the elements of the data structure one per mnemond (with possible wrap-around to the beginning of the linear array) or allocating the data structures to one or a predetermined number of mnemonds. When a data structure is to be located in memory through cueing precise correspondences between data stored and cueing data entered identify the unique location or locations of said matching data; when a data structure is to be recovered from memory identification of its location permits recovery of data in that location or in that location and succeeding locations. One or more arrays, of differing lengths so that the allocation of data elements is also differing, is collected to form a memory bank. One or more memory banks operates under the control of a central control unit.

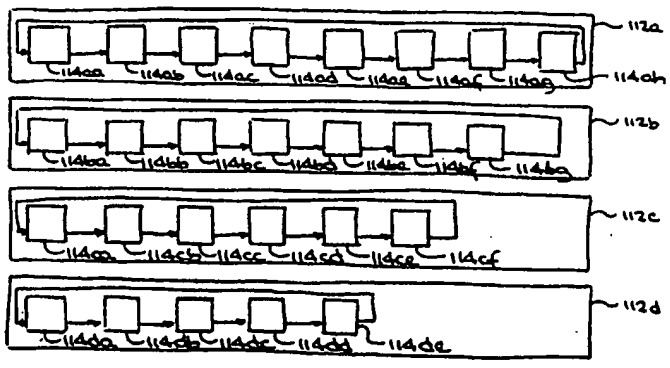


Figure 1.

EP 0 274 426 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 88 30 0074

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.4)
A	US-A-4 065 756 (G. PANIGRAHI) * Whole document *	1-3,5, 12,14-21	G 06 F 15/40
A	GB-A-2 182 789 (TEXAS INSTRUMENTS LTD) * Abstract; page 1, line 1 - page 2, column 2, line 111; claims *	1-3,5, 12,14-21	
A	EP-A-0 139 817 (SIEMENS AG) * Abstract; claims *	1-3,5, 12,14-21	
A	IBM TECHNICAL DISCLOSURE BULLETIN, vol. 10, no. 10, March 1968, page 1464, Armonk, US; L.M. TERMAN: "Shift register associative memory" * Whole document *	22-24	

TECHNICAL FIELDS SEARCHED (Int. Cl.4)

G 06 F

~~The present search report has been drawn up for all claims~~

Place of search	Date of completion of the search	Examiner
THE HAGUE	31-03-1992	FOURNIER C.D.J.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0601)

European Patent Office

EP 88 30 0074

CLAIMS INCURRING FEES

The present European patent application comprised at the time of filing more than ten claims.

☐ All claims fees have been paid within the prescribed time limit. The present European search report has been drawn up for all claims.

☐ Only part of the claims fees have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims and for those claims for which claims fees have been paid,

namely claims:

☐ No claims fees have been paid within the prescribed time limit. The present European search report has been drawn up for the first ten claims.

LACK OF UNITY OF INVENTION

The Search Division considers that the present European patent application does not comply with the requirement of unity of invention and relates to several inventions or groups of inventions.
namely:

See Sheet B.

☐ All further search fees have been paid within the fixed time limit. The present European search report has been drawn up for all claims.

☐ Only part of the further search fees have been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the inventions in respect of which search fees have been paid,

namely claims:

☒ None of the further search fees has been paid within the fixed time limit. The present European search report has been drawn up for those parts of the European patent application which relate to the invention first mentioned in the claims. 1-24

namely claims:

European Patent Office

EP 88 30 0074 -B-

LACK OF UNITY OF INVENTION

The Search Division considers that the present European patent application does not comply with the requirement of unity of invention and relates to several inventions or groups of inventions, namely:

1. Claims 1-24: Memory bank, array, mnemonds.

2. Claims 25-37: Data format.

3. Claims 38-42: Program shift register.

4. Claims 43-45: AXOND gate.

5. Claim 46: Pattern detector circuit.

6. Claims 47-48: Translation circuit.